DECISION DIRECTED AND CONSTANT MODULUS ALGORITHMS DERIVED AND EVALUATED FOR MULTICARRIER SYSTEMS

THESIS

Nicholas L. Linnenkamp, Second Lieutenant, USAF

AFIT/GE/ENG/06-36

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government.

AFIT/GE/ENG/06-36

DECISION DIRECTED AND CONSTANT MODULUS ALGORITHMS DERIVED AND EVALUATED FOR MULTICARRIER SYSTEMS

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Electrical Engineering

Nicholas L. Linnenkamp, BS

Second Lieutenant, USAF

March 2006

AFIT/GE/ENG/06-36

DECISION DIRECTED AND CONSTANT MODULUS ALGORITHMS DERIVED AND EVALUATED FOR MULTICARRIER SYSTEMS

Nicholas L. Linnenkamp, BS
Second Lieutenant, USAF

Approved:

_____/Signed/_____ _____
Dr. Richard K. Martin (Chairman) date

_____/Signed/_____ _____
Dr. Richard A. Raines (Member) date

_____/Signed/_____ _____
Dr. Michael A. Temple (Member) date

AFIT/GE/ENG/06-36

Abstract

The purpose of this research is to derive and examine blind adaptive algorithms for equalizing multicarrier (MC) communication systems and analyze how they perform under varying environmental conditions and parametric variations, focusing on equalizers set in cascade with the channel. Two well-accepted and widely-known cost functions, Decision Directed (DD) and Constant Modulus (CM), are applied to the MC signal structure, and gradient descent algorithms based on both DD and CM functions are derived, analyzed and compared. Comparison of the new algorithms, Multi Carrier Decision Directed (MCDD) and Multi Carrier Constant Modulus (MCCM), focuses on detailing how each algorithm performs when the factors of noise power and symbol synchronization are varied. Additionally, a Frequency Domain Equalizer (FEQ) is developed and employed to de-rotate and resize the output symbol constellation. Both MCDD and MCCM are compared against other blind and trained adaptive MC equalization algorithms in the areas of bit error rate (BER) vs. signal to noise ratio (SNR) and BER vs. synchronization delay. Results are presented showing that MCDD and MCCM perform worse than a MC Trained (MCT) approach but better than both Multicarrier Equalization by Restoration of Redundancy (MERRY) and Carrier Nulling Algorithm (CNA).

To Jennifer, for your dedication and support

Acknowledgments

I would like to express my sincere appreciation to my faculty advisor, Dr. Richard Martin, for his guidance and support throughout the course of this thesis effort as well as his understanding and patience for dealing with my selective memory. I would, also, like to thank my thesis committee, Dr. Richard Raines and Dr. Michael Temple, for their support in helping in the revision process. I would also like to thank my sponsor, Mr. Jim Stephens, from the Air Force Research Laboratory for taking interest in my work.

Nicholas L. Linnenkamp

Table of Contents

Page

List of Figures

Figure Page

List of Tables

DECISION DIRECTED AND CONSTANT MODULUS ALGORITHMS DERIVED AND EVALUATED FOR MULTICARRIER SYSTEMS

I. Introduction

Background

Equalizers have grown in importance over the last few decades due to the rise in number of communication systems and have application over a wide variety of communication mediums. Example systems include those based on communications over the telephone infrastructure built on the ITU V.XX standards, over broadband coaxial infrastructure used by cable companies for cable-television and high-definition television (HDTV), and over wireless communication channels found in cell phones along with other terrestrial wireless channels. Blind adaptive algorithms are algorithms that track the channel without having to exchange any additional training information or having prior knowledge about the channel. Blind adaptive algorithms are attractive to many applications since training information doesn't have to be sent reducing algorithm overhead. Although trained adaptive algorithms take less time to equalize the channel and have lower processing cost they are prohibitive in situations involving broadcast mediums or channels that are already bandwidth constrained.

Broadcast mediums such as cell phones and cable television tend to have a one-to-many type distribution scheme. A cell phone that is powered on needs to equalize to the channel in order to receive information from cell-tower and in this case may not be able to tell when a training sequence begins. It is possible to start a new communication with each client that tries to connect in order to train them to the channel but if there are a lot of clients this can become bandwidth costly. In addition, the act of sending periodic training sequences decreases available bandwidth. Similar results can be found in the wire broadcast mediums of cable television. In these situations it would be of great benefit to have blind equalization where the clients were able to train to the channel without involving the server side. Blind equalization removes the need to send periodic training sequences at the cost of an increase in computation on the client and slower convergence of the equalizer.

For bandwidth constrained channels where the needs of the client use up all available bandwidth resources any methods that can increase the carrying capacity of the channel for the client is a marked improvement. An application that is data transfer starved, as is often the case when using modems that run over the telephone lines, would benefit greatly from blind equalization due to the bandwidth freed from not using bits for training.

Equalization has found wide spread use in multicarrier (MC) communication systems by helping to reduce or remove channel effects such as inter-carrier and inter-symbol interference (ICI, ISI), see Johnson [1] and Martin [2] for a treatment of ISI and ICI. ICI is caused when the orthogonal signals in a MC system are spread in frequency so that they are no longer orthogonal and degrade the ability to distinguish one signal

from others. ISI is when the channel causes one transmitted symbol to spread in time into another symbol. When symbol rates are high ISI becomes a concern because of the small spacing between the signals. MC communication systems, such as Satellite Radio, HDTV, wireless LANs and digital subscriber lines (DSL), make use of a cyclic prefix (CP) inserted before each symbol so that interference can be removed, see [3], [4], [5], and [6] for treatment of Digital Audio Broadcast (DAB), Digital Video Broadcast, Wireless LANs and DSL. The CP is introduced so that the convolution of the channel with the data looks circular. If the length of the CP does not exceed the length of the channel then interference occurs. Although the CP can be made arbitrarily long and remove ISI and ICI, it involves the transmission of redundant bits, reducing available bandwidth for transmitting information. In order to help keep the length of the CP as short as possible and assist in removing interference, a time-domain equalizer (TEQ) can be placed after the channel. Using a TEQ to shorten the channel, effectively reducing the channel memory, allows for a shorter CP which increases information transmission capacity [2].

TEQs for MC systems can be implemented by either placing one filter after the channel (in cascade) or by a bank of filters tone-by-tone (per tone. Using one filter in cascade allows global equalization over all tones, which aims to reduce interference but typically does not focus on maximizing bit rate. On the other hand, per-tone equalization has the ability to maximize throughput on each tone and can be used to optimize the bit rate for the system. Although per-tone equalization increases bit rate for the system it also requires an additional equalizer for every tone each of which requires adapting vastly increasing the computational complexity. A cascade implementation is attractive

because only one equalizer needs to be adapted). See [7] for a discussion of per-tone equalization and [2] for additional discussion of per-tone and cascade equalizers.

Equalizers can either be set statically or adaptively. Statically set TEQs can be used when the channel doesn't change at all or very infrequently whereas adaptive TEQs are used when the channel is changing. A static TEQ can require a large one-time upfront computational cost in computing the channel shortener but then the system runs without significant additional equalizer computation. Adaptive filters have low up front computational cost but require some time to converge and ongoing update computations.

Problem Statement

Blind adaptive equalization algorithms developed for single carrier channels do not directly apply to MC systems. Changes in the receiver structure, such as a Fourier transform and Frequency Domain Equalizer, contribute to the necessity of modifying these algorithms. It would be advantageous to have several of the single carrier blind adaptive equalization algorithms modified so that they take into account the MC receiver structure. The new algorithms would be based on the cost functions of the single carrier algorithms but include the complexity of the new receiver structure.

Research Objectives/Questions/Hypothesis

The objective of this research is to take two well known cost functions, decision directed and constant modulus, and modify them to take into account a MC receiver structure. Decision directed and constant modulus are the two blind algorithms most often deployed when blindly updating single-carrier equalizers, and examination of how

they work gives insight into what can be expected when they are used to update MC TEQs. Both form updates from information about the symbol set.

In a simplified example of decision directed, if the symbol set consists of 1 and -1, then by observing the output and comparing it to a decision boundary of zero, steps can be made in the right direction of the proper TEQ filter weights. More complicated symbol sets require more complicated decision regions but the Decision Directed (DD) cost function remains the same. The DD cost function is similar in nature to a trained cost function except you have the possibility of noise causing an error in the determination of the symbol and of phase ambiguity.

In a simplified example of Constant Modulus (CM) using 1 and -1, if the output is squared the result should be similar to the square of the input. Squaring the input gives us a constant modulus of 1 and any deviations from that in the output will be penalized driving a solution for the TEQ. More complicated symbol sets can be handled by creating decision moduli for the symbol set and comparing the closest to the modulus of the received symbol, but in practice usually only one modulus is used, even for multi-level quadrature amplitude modulation (QAM). See [1] to see an in-depth look at CM for single carrier systems.

Both update rules have strengths and weaknesses. Traditional DD tends to drive away from a good solution when bit error rates (BERs) are over 10-15%, which is often the case just after the device is turned on, also known as a cold start while CM relies on higher order statistics requiring additional computation. Often in deployed blind adaptive equalizers it is found that CM is used initially when adapting the equalizer, and once BERs are low enough, DD is employed.

The term symbol synchronization has a different meaning in the context of shortening algorithms then when generally referring to communication systems. When referring to general communication systems, symbol synchronization usually refers to when the receiver samples to determine an individual symbol. Symbol synchronization in the context of this thesis and MC equalization algorithms is when a receiver is first powered on and has not equalized the channel, the receiver must guess at which collection of symbols to input to the Fourier transform [8]. A deviation from perfect symbol synchronization is measured in whole symbols. Since a good guess at symbol synchronization results in better channel equalization it would be convenient to know which range of values are good and which ones provide poor performance. Both DD and CM have been studied in single carrier situations involving white, Gaussian noise and perfect symbol synchronization. Non-perfect symbol synchronization has not been studied in depth with these algorithms and can be shown to have a substantial effect on algorithm performance. Neither of the two algorithms has been derived for MC systems. Once the MC versions are derived, convergence rates and computational cost need to be evaluated. It is assumed that the MC versions of the algorithms will have similar characteristics as the single-carrier versions of the algorithms because they are based on the same cost function.

Research Focus

The focus of this thesis is to derive and examine blind adaptive algorithms for equalizing MC communication systems and analyze how they perform under varying

system constraints focusing on equalizers set in cascade with the channel. Two well-accepted and widely-known cost functions, Decision Directed (DD) and Constant Modulus (CM), are applied to the MC signal structure, and gradient descent algorithms based on both DD and CM are derived, examined and compared. The comparison of the new algorithms, Multi Carrier Decision Directed (MCDD) and Multi Carrier Constant Modulus (MCCM) will focus on detailing how each algorithm performs when the factors of noise power and symbol synchronization are varied. Additionally, Frequency Domain Equalizer (FEQ) adaptive algorithms are developed and employed to de-rotate and resize the output symbol constellation. A differential encoder is used on the input sequence so that a static phase rotation can be negated by a differential encoder on the output sequence.

Implications

Having these algorithms available will enable professionals and researchers to implement these algorithms in a variety of systems. In addition, these algorithms will hopefully become a basis for evaluating new blind and MC algorithms as DD and CM have been for single-carrier systems. Both MCDD and MCCM performance will be shown against other existing blind adaptive channel shortening algorithms and comparisons will be made.

Preview

The remainder of this thesis outlines the system model, background/literature search, algorithm development, results and analysis, and finally concludes. Chapter 2

focuses on the system model, introducing an orthogonal frequency division multiplexed system as well as introducing terminology used throughout the rest of the paper. Next, Chapter 3 provides a body of previous work to develop a framework of reference in which the new algorithms are compared against and classified. Next, the document moves into Chapter 4 introducing the algorithms being developed, providing the mathematical framework under which the algorithms are generated, and discussing their computational complexity. Chapter 5 discusses system operation and shows the effect of a trained adaptive equalization algorithm while extrapolating results for the remainder of the adaptive algorithms. Lastly, Chapter 6 provides simulation results for the algorithms being derived as well as several other algorithms found in the background and literature search and concludes with areas of future research.

II. System Model

Notation

This section lays out the framework for the notational context and conventions used in the rest of the thesis document. Notation for vectors, their transpose and conjugates, matrices, scalars and other conventions will be given.

Column vectors are represented by a lowercase or uppercase designator with an over-bar, for example:

$$\bar{r} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ \vdots \\ r_k \end{bmatrix}$$

Likewise, designators without over-bars are considered matrices and are generally complex.

OFDM symbols are generally created in the frequency domain and transformed into the time domain in the transmitter. Uppercase variable names designate the frequency-domain and lowercase variable names designate time-domain.

Vector transpose is denoted by a superscripted T on the matrix or designator and can save space when displaying long column matrices, for example:

$$\bar{r} = \begin{bmatrix} r_1 & r_2 & r_3 & \cdots & r_k \end{bmatrix}^T \text{ OR } \bar{r}^T = \begin{bmatrix} r_1 & r_2 & r_3 & \cdots & r_k \end{bmatrix}$$

Vector conjugation is denoted by a superscripted conjugate symbol * on the matrix or designator, for example:

$$\bar{r}^{*} = \begin{bmatrix} r_1 & r_2 & r_3 & \cdots & r_k \end{bmatrix}^{*} = \begin{bmatrix} r_1^{*} & r_2^{*} & r_3^{*} & \cdots & r_k^{*} \end{bmatrix}$$

The Hermitian is denoted by a superscripted H and is the equivalent transpose and conjugation of a matrix,

$$\bar{r}^{H} = \begin{bmatrix} r_1^{*} & r_2^{*} & r_3^{*} & \cdots & r_k^{*} \end{bmatrix}^{T}$$

The system model will be working on blocks of input symbols. Each block of input symbols is given a number referring to that input block. When referring to the block of symbols a vector will have a parenthetical identifier denoting which block is in reference, for example:

$$\bar{S}(k) = \begin{bmatrix} S_{Nk+1} \\ S_{Nk+2} \\ S_{Nk+3} \\ \vdots \\ S_{Nk+N} \end{bmatrix}$$

Here $\bar{S}(k)$ is a vector of N input symbols taken from a larger set of input symbols where k refers to the block in reference; hence the variable k is an argument to the vector.

The gradient operator ∇, as often seen in multidimensional calculus, is applied to the adaptive equalizers in this thesis as to other gradient descent algorithms. The gradient operator will be taken with respect to our time domain equalizers (TEQs) and denoted by a subscript indicating which TEQ it is being taken with respect to, such as $\nabla_{\bar{w}_1}$, the gradient with respect to TEQ w_1. The gradient operator can equivalently be though of as a column vector of partial derivatives with respect to each tap of the TEQ. The following formula introduces the column vector notation for the gradient operator,

$$\nabla_{\underline{w}_1} = \begin{bmatrix} \dfrac{\partial}{\partial w_1(1)} \\ \dfrac{\partial}{\partial w_1(2)} \\ \vdots \\ \dfrac{\partial}{\partial w_1(L_w)} \end{bmatrix}$$

Where the values in parenthesis indicate the tap of the TEQ the partial derivative is being taken with respect to.

The Fourier transform is an integral part of MC systems and notation needs to be developed to represent a Fourier transform operation. Taking the Fourier transform can be seen as multiplication by a discrete Fourier Transform (DFT) matrix. The DFT matrix can be derived easily by taking the Fourier transform of an identity matrix. The notation for the Fourier transform is denoted by a $\mathcal{F}$ and the inverse Fourier transform by a $\mathcal{F}^{-1}$.

When referring to a portion of a matrix which will often be the case when deriving the blind adaptive algorithms, the first subscripted index will be used to denote row and the second to denote column, for example, $\mathcal{F}_{l,i}$ refers to the l^{th} row and i^{th} column of the DFT matrix. If only one subscript is given it refers to all entries of the row of the matrix it is augmenting. If there is only one entry in the row, as is the case for column vectors, it refers to only that entry.

System Under Test

The system shown below is typical of a fractionally spaced or multi-antenna multicarrier (MC) system with fixed TEQs. Although not shown, in order to make the TEQs adaptive, a feed back loop is inserted within the receiver (as opposed to feedback

to the transmitter) after the frequency domain equalizer (FEQ) so that the MCDD and MCCM update rules can be applied to adapt the TEQs. It is the focus of this study to derive the update rules and to analyze the performance under varying conditions. The following figure shows in block diagram form the system under test.

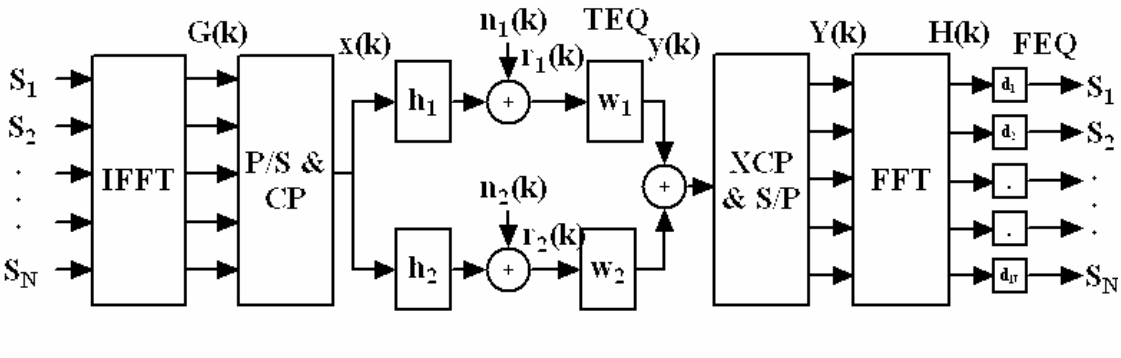

Figure 1. System Under Test

The system model includes a Fractionally Spaced Equalizer (FSE) and is implemented by using two channels and two noise sequences. FSEs can be implemented by a receiver in several ways, such as installing two antennas in the receiver or by over-sampling the same receiver. Either way, both received sequences look as if they have undergone different channels and different additive noise sources from each other.

The following discussion steps through the system under test and describes in detail the operations being performed at each juncture in the diagram. This system will be the basis for all the algorithms in the following sections motivating a detailed and thorough description.

Figure 1 represents in block diagram format the system under test. On the left, as input to the system, input data is broken into blocks of N complex symbols or less depending on if null carriers are used or not,

$$\bar{S}(k) = \begin{bmatrix} S_{Nk+1} \\ S_{Nk+2} \\ S_{Nk+3} \\ \vdots \\ S_{Nk+N} \end{bmatrix} \quad (1)$$

where N is the block size and k indicates the block in reference. The input symbol set is often quadrature amplitude modulated (QAM) with the size of the constellation dependent on the type of application. For this study, 4-QAM will be used.

Since one of the algorithms detailed in the Results and Analysis section is the Carrier Nulling Algorithm (CNA) a brief discussion of null carriers and how they are implemented should be discussed. CNA assumes that there are null carriers on several channels of the MC communication system. Null carriers are defined as carriers that only transmit the symbol zero. These null carriers can be used for any number of reasons but the primary motivating factor is for guard bands. Guard band implementation is done by setting the edge tones of a MC communication system to null carriers to ensur that systems using frequencies near the system don't accidentally drift or encroach upon the system don't interrupt proper operation.

In the case where null carriers are used not all of the N symbols will have complex 4-QAM data on them. After the input symbols are put through the Inverse Fast Fourier Transform (**IFFT**),

$$\bar{G}(k) = \mathcal{F}^{-1}\bar{S}(k) \quad (2)$$

they are input into the parallel/serial and cyclic prefix block (**P/S & CP**). The output sequence, x(k), is then found to have length $M = N + v$, where v is the length of the cyclic prefix. M is considered the transmission block size.

Thus the last v symbols are appended to the front of the sequence,

$$\bar{x}(k)=\left[G(Nk-v+N) \quad G(Nk-v+N) \quad \cdots \quad G(Nk+N) \quad G(Nk+1) \quad G(Nk+2) \quad \cdots \quad G(Nk+N)\right]^T \tag{3}$$

and transformed from parallel to serial. If the subscript *i* is used as an index into the block then when the channel h_1 and h_2 convolutes x(k) and noise is added, the result is $r_1(Mk+i)$ and $r_2(Mk+i)$. Here $r_1(Mk+i)$ and $r_2(Mk+i)$ are representative of either sequences received from a single oversampled antenna or possibly two spatially separate antennas. $r_1(Mk+i)$ and $r_2(Mk+i)$ shown below,

$$r_1(Mk+i) = \sum_{j=1}^{L_h} h_1(j)x(Mk+i-j+1) + n_1(Mk+i) \tag{4}$$

$$r_2(Mk+i) = \sum_{j=1}^{L_h} h_2(j)x(Mk+i-j+1) + n_2(Mk+i) \tag{5}$$

After the equalization filter w, the output $y_1(Mk+i)$ and $y_2(Mk+i)$ are

$$y_1(Mk+i) = \sum_{j=1}^{L_w} w_1(j)r_1(Mk+i-j+1) \tag{6}$$

$$y_2(Mk+i) = \sum_{j=1}^{L_w} w_2(j)r_2(Mk+i-j+1) \tag{7}$$

These results are then summed,

$$y(Mk+i) = y_1(Mk+i) + y_2(Mk+i) \tag{8}$$

The next block takes the stream from serial back to parallel format and removes the cyclic prefix. At this stage, Δ, the synchronization delay, is accounted for since it represents the uncertainty of where the receiver should break the input into blocks.

$$\bar{Y}(k) = \begin{bmatrix} y(Mk+v+1+\Delta) \\ y(Mk+v+2+\Delta) \\ \vdots \\ y(Mk+M+\Delta) \end{bmatrix} \tag{9}$$

After the cyclic prefix is removed and the blocks are back in parallel format then each block is put through a Fast Fourier Transform (**FFT**),

$$\overline{H}(k) = \mathcal{F} \cdot \bar{Y}(k) \tag{10}$$

Lastly, the output of each tone from the FFT is run through a one-tap frequency domain equalizer (FEQ) to de-rotate and scale the output symbol so that the output again looks like a 4-QAM symbol set.

The effective channel c, is defined as the sum of the convolutions of each channel with its respective TEQ, $\bar{c} = \overline{c_1} + \overline{c_2}$, where $\overline{c_1} = \overline{h_1} * \overline{w_1}$ and $\overline{c_2} = \overline{h_2} * \overline{w_2}$.

Summary

One of the most important things to note about the system model is the symbol synchronization delay. A goal of this study is to characterize the algorithms under test according to how synchronization delay affects their performance so understanding how

this delay affects the receiver can benefit analysis in the subsequent chapters. Because there are only N samples in every block and the transmitter adds a CP of v samples then the synchronization delay can only vary between 0 and $N+v$ before repeating. It is expected then that a bit error rate (BER) plot vs. synchronization delay will be periodic, that is the BER for a delay of $N+v+1$ should be the same as for a delay of 0.

After giving a thorough discussion of the system model, the next chapter outlines the background and literature search of relevant blind adaptive equalization algorithms for both single carrier and MC systems.

III. Background and Literature Search

Overview

While there are many trained TEQ designs, some of the most understood and deployed TEQ systems are focused on three different cost functions: Mean Squared Error (MSE), Shortening SNR (SSNR), and Geometric SNR (GSNR). There are considerably less single-carrier blind equalization designs, among which the most deployed use Decision Directed (DD) and Constant Modulus (CM) cost functions. Blind multicarrier TEQ designs of importance include Multicarrier Equalization by Restoration of RedundancY (MERRY), Sum-Squared Auto-Correlation Minimization (SAM), and the Carrier Nulling Algorithm (CNA). These approaches, while not all inclusive, form a large basis from which to work and understand how blind adaptive Multi Carrier Decision Directed (MCDD) and Constant Modulus (MCCM) should be developed.

Minimum MSE

Minimum MSE (MMSE) was first introduced by Falconer and Magee [9] as a method to shorten the impulse response of a single carrier channel. The motivation to shorten the channel was drawn from a desire to reduce the complexity of the Viterbi decoding algorithm, since its complexity depends on channel memory. See [23] for implementation and complexity dependencies of Viterbi coding as well as [2] and [9] for treatment of MMSE. Later, Chow and Cioffi [10] applied MMSE to multicarrier systems. Figure 2 shows in block diagram form how the MMSE error signal is formed, where TIR is the target impulse response of length $\nu+1$.

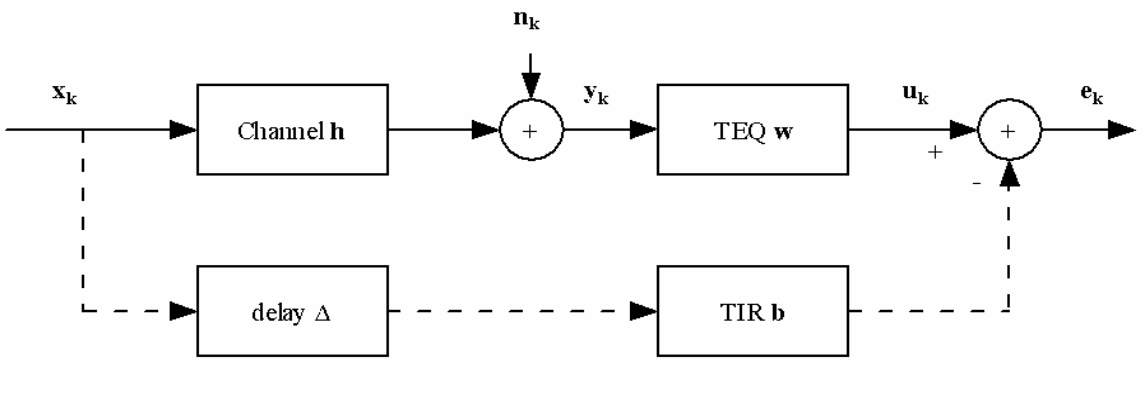

Figure 2. Block Diagram for MMSE Channel Shortener

A cost function, based upon the error signal e_k, is developed and minimized. Knowledge of the bits sent is required to be able to determine the error which is why this is classified as a trained TEQ. The cost function, *J*, then becomes a function of both the TIR and the TEQ.

$$J(w,b) = E\left\{(e_k)^2\right\} = E\left\{\left(\overline{w}^T \overline{y_k} - \overline{b}^T \overline{x}_{k-\Delta}\right)^2\right\} \qquad (11)$$

Hence it is called a MMSE design because it attempts to minimize the mean squared error.

Maximum Shortening SNR

MSSNR is a trained TEQ design proposed by Melsa, Younce, and Rohrs in 1996 [11] in which they attempt to constrain the channel impulse response (CIR) so that the result mostly resides in $\nu+1$ consecutive samples, called a window.

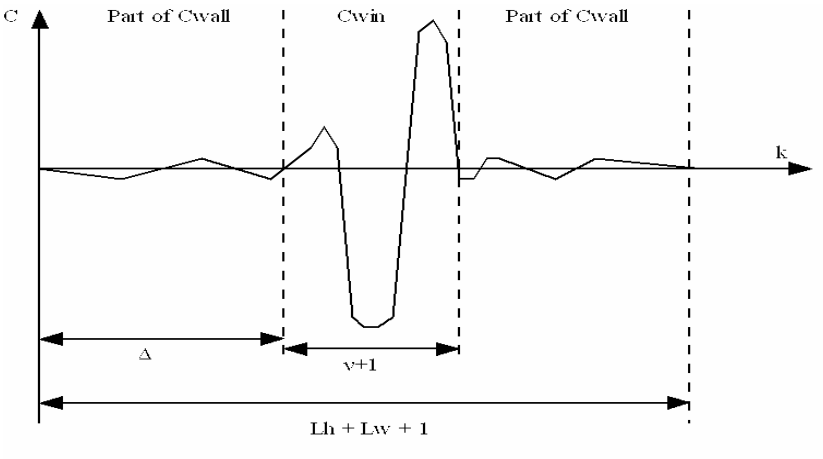

Figure 3. Picture Showing Window and Wall of CIR

By minimizing the fraction of energy outside of that window, or equivalently by maximizing the fraction of energy inside the window, the CIR is shortened. With a shorter effective channel the CP can be shorter and the effective bit rate increases. The MSSNR method requires that you have knowledge of the channel in order to minimize/maximize channel energies which is why MMSNR is classified as a trained method since training bits need to be sent across the channel in order to estimate the channel. The cost function attempts to maximize the ratio of windowed energy to non-windowed or equivalently the overall channel energy to non-windowed energy.

Maximum Geometric SNR

While MMSE and MSSNR will increase bit rate, neither functions have bit rate as part of the cost. MGSNR, first proposed by Al-Dhahir and Cioffi in [12] and [13], is designed to maximize the bit rate of a MC system by observing the SNR on each subchannel of the system and weighting the results based upon the amount of bits per sample transmitted on that subchannel. A cost function called the Geometric SNR

(GSNR) is formulated from the product of the subchannel SNRs and maximized. While per-tone equalization can be used to maximize the bit rate on a single subchannel, MGSNR attempts to globally maximize bit rate with one TEQ across all channels. Drawbacks to MGSNR are that it is only an approximate and even though it uses only one TEQ across all channels it still suffers from high computational cost.

Decision Directed (DD) and Constant Modulus (CM) Algorithms

All of the above designs rely on either knowing the transmitted bit sequence or on knowledge of the channel. While these designs are adequate for most systems, some systems and applications require the use of blind TEQs in order to achieve optimal performance.

Two single carrier blind equalizer designs, DD and CM, have a long history dating back to the 1970's and 1980's, see [1] for history and a more complete treatment of CM as well as [14] for discussion of both DD and CM. The focus of DD and CM is on equalizing the channel using information about the transmitted symbol set to derive an error function and performing a gradient descent along that error function. While these single carrier approaches have been around for a while, more recent work in blind TEQ designs tend to exploit additional information about the transmitted sequence or specific characteristics between the transmitter and receiver.

Multicarrier Equalization and Restoration of Redundancy

An example of a system which uses additional information other than the symbol set is the MERRY algorithm proposed by Martin, Balakrishnan, Sethares, and Johnson in

2002 [8] which uses information about the CP to equalize the channel. MERRY creates a cost function which attempts to minimize the difference between the last sample in the CP with the last sample in the symbol since they should be equal.

$$J_{\delta} = E\left\{ \left| y(v+\Delta) - y(v+N+\Delta) \right|^2 \right\} \qquad (12)$$

This cost function requires knowledge of the symbol synchronization, Δ, since it has substantial effect on the cost function. By taking advantage of the redundancy of the transmitted bits it is possible to use this information on the receiving end in a blind fashion to equalize the channel.

Sum-Squared Auto-Correlation Minimization

Whereas the MERRY algorithm bears resemblance to a blind MMSE design since both use a first order difference, the SAM algorithm bears resemblance to the MSSNR design where it tries to minimize the energy outside of the a window of ν + 1 samples, but in a blind fashion.

The SAM algorithm, developed by Balakrishnan, Martin, and Johnson in [15], formulates a cost function built on the auto-correlation of the received signal based upon the fact that autocorrelation values that are more than ν samples apart should be as small as possible in order for the channel to be constrained to the window. By rewriting the channel correlation as functions of the received signal as well as second and fourth order products of the noise variance then observing that the noise products can be neglected, a blind cost function is developed.

$$J_{SAM} = \sum_{l=v+1}^{L_c} E\left\{ \left| y(n)y(n-l) \right|^2 \right\} \qquad (13)$$

This blind cost function is then applied in a gradient descent fashion in order to implement a blind adaptive TEQ. Another paper derived from the work of SAM is shown in [16] and is worth noting here.

Carrier Nulling Algorithm

Lastly, CNA, presented by de Courville, Duhamel, Madec, and Palicot in [17], as well as Romano and Barbarossa in [18], attempts to exploit additional information about the transmitter structure. Since in typical MC systems some carriers only transmit zero, they have been dubbed null carriers. Since it is known *a priori* that these carriers will be zero then a cost function based on the null carriers can be formulated,

$$J_{CNA} = \sum_{i \in \bar{S}_n} E\left\{ \left| G(k) \right|^2 \right\} \qquad (14)$$

where $\bar{S}_n$ represents the set of null carriers, and $G(k)$ is the DFT output on tone *k*. By taking advantage of this information, a low-complexity, adaptive minimization procedure is developed, since the cost function is of low complexity and the transmitted data should not affect the null carriers.

Summary

These systems, particularly those focusing on blind adaptive TEQs, provide the motivation for developing and analyzing blind adaptive update rules for MCDD and MCCM. It is from this common framework that the system model was developed for

implementation and analysis of both MCDD and MCCM algorithms. Although not directly sourced, [19] and [20] give good background information on blind equalization while [21] suggests exploiting symmetry in the TEQ. Finally, [22] is an additional paper on bit rate maximization using a single TEQ

The next chapter delves into deriving and implementing the update algorithms used to adapt the TEQs of the system model. Using a gradient descent approach several trained and blind algorithms are proposed. These algorithms are then shown to adequately adapt to the channel and shorten the effective response. In addition to adaptive TEQ algorithms, algorithms for adapting the FEQ are derived as well since if the TEQ changes, the FEQ must change as well.

IV. Algorithm Development

Overview

The algorithms in this chapter focus both on updating the Time Domain Equalizer (TEQ) as well as updating the Frequency Domain Equalizer (FEQ) in multicarrier systems. The first two algorithms to be derived are the Multicarrier Trained (MCT) and the Multi Carrier Decision Directed (MCDD). These two algorithms will be derived together since they are so closely related that their derivation is very similar. Next, the derivation for the Multi Carrier Constant Modulus algorithm is shown. Finally, a trained FEQ, the Least Mean Squares (LMS) – FEQ, as well as a blind FEQ, the Decision Directed LMS-FEQ, are derived.

The algorithms presented in this chapter are the major contribution to knowledge of trained and blind adaptive equalization algorithms. The next chapter focuses on exercising these algorithms and comparing them against other MC equalization algorithms.

Multi Carrier Decision Directed

The MCDD algorithm formulates an error function based on a first-order difference between a decision directed function and the output symbol. When the gradient with respect to the TEQ is taken over the cost function, the magnitude squared of the error function, the result is first order. The linear response of the gradient of the cost function ensures that steady, predictable progression towards a minimum is achieved. While not

globally convergent in the single carrier case, for cases where $\overline{w}$ is close to its global minimum fast convergence is expected. In contrast, a second-order error function results in a fourth-order cost function and ultimately the gradient of the cost function is third-order. A higher-order cost function results in faster convergence than single-order when farther away from a minimum and slower convergence when close to the minimum.

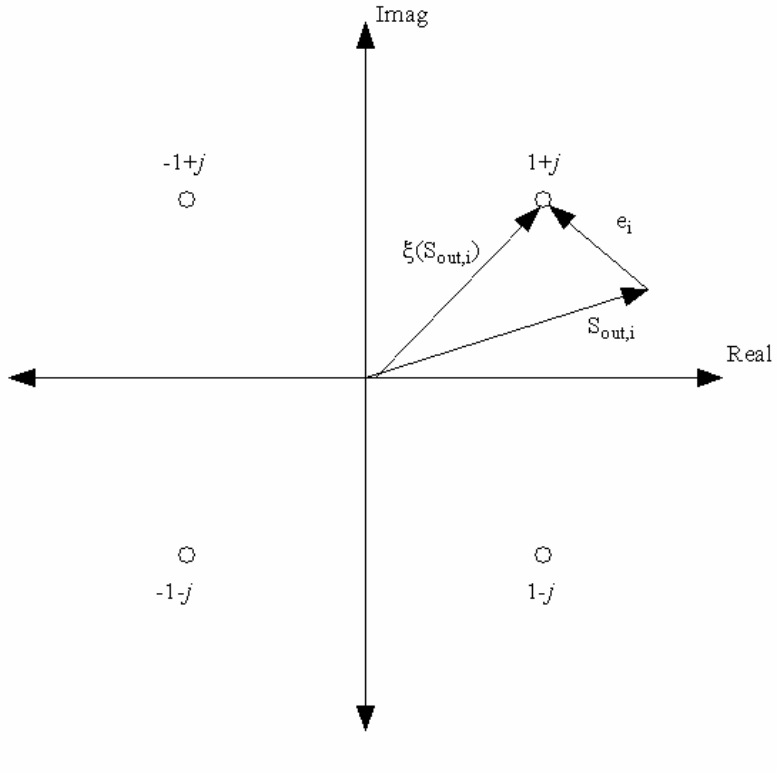

Figure 4. Illustration of Decision Directed (DD) Error e_i

Figure 4 illustrates how the error, $e_{i,}$ is generated and the concept of the decision directed function. Here ξ is the decision direction function whose output is the closest symbol in the communication system's constellation. When the BER is high DD does not perform very well due to the fact that the decision function reports the closest symbol. When an

error happens the result is a smaller error which points in a direction away from the correct input symbol. This effect causes the single carrier DD algorithm to converge to an incorrect setting when BERs are high. Most of these results, such as convergence speeds and globally/not globally convergent, are expected to hold for MCDD since it is derived from a similar cost function.

As in the single carrier case for a DD update algorithm the cost function remains similar for each of the MC output taps. If *i* designates the output of interest then the MCDD cost function can be written,

$$J_{MCDD} = \sum_{i=1}^{N} J_{MCDD,i} = \sum_{i=1}^{N} e_i e_i^* \qquad (15)$$

Where e is the error and e* is error compliment

$$e_i = \xi\left(S_{i,out}\right) - S_{i,out} \qquad (16)$$

There will be an error for each of the tones in the MC communication system thus the object is to reduce the sum of the cost functions for all of the tones. By taking the gradient with respect to the TEQ filters w_1 and w_2, represented by $\nabla_{\underline{w_1}}$ and $\nabla_{\underline{w_2}}$, on each tone an update rule is derived that will ensure that the cost function moves towards a minimum.

$$\nabla_{\underline{w_1}} J_{MCDD\,i} = \nabla_{\underline{w_1}}\left(e_i e_i^*\right) \qquad (17)$$

$$\nabla_{\underline{w_1}} J_{MCDD\,i} = \left(\nabla_{\underline{w_1}} e_i\right) e_i^* + e_i\left(\nabla_{\underline{w_1}} e_i^*\right) \qquad (18)$$

Looking just at $\nabla_{\underline{w_1}} e_i$ for the first TEQ filter w_1,

$$\nabla_{\underline{w_1}} e_i = \nabla_{\underline{w_1}} \xi - \nabla_{\underline{w_1}} S_{i,out} \tag{19}$$

Since ξ always resolves to a constant then the gradient of ξ with respect to the TEQ is always zero. $S_{i,out}$ defined here from the system under test,

$$S_{i,out} = FEQ_i \sum_{l=1}^{N} \mathcal{F}_{i,l} \sum_{j=1}^{Lw} (w_1(j) r_1(Mk + l - j + 1) + w_2(j) r_2(Mk + l - j + 1)) \tag{20}$$

Now examining $\nabla_{\underline{w_1}} S_{i,out}$,

$$\nabla_{\underline{w_1}} S_{i,out} = \nabla_{\underline{w_1}} FEQ_i \sum_{l=1}^{N} \mathcal{F}_{l,i} \sum_{j=1}^{Lw} (w_1(j) r_1(Mk + l - j + 1) + w_2(j) r_2(Mk + l - j + 1) \tag{21}$$

Here, $\mathcal{F}_{l,i}$ is the element (*i*,*l*) of the DFT matrix. Since the FEQ_i and $\mathcal{F}_{l,i}$ do not depend on w_1 they can be treated as constants with respect to the gradient, $\nabla_{\underline{w_1}}$. Additionally, the terms containing $w_2(j)$ go to zero when the gradient is taken with respect to w_1.

$$\nabla_{\underline{w_1}} S_{i,out} = FEQ_i \sum_{l=1}^{N} \mathcal{F}_{l,i} \nabla_{\underline{w_1}} \sum_{j=1}^{Lw} w_1(j) r_1(Mk + l - j + 1) \tag{22}$$

When moving the gradient inside the sum a little more work has to be done to progress further. Examining the sum more closely,

$$\nabla_{\underline{w_1}} \sum_{j=1}^{Lw} w_1(j) r_1(Mk + l - j + 1) =$$

$$= \begin{bmatrix} \dfrac{\partial}{\partial w_1(1)} \\ \dfrac{\partial}{\partial w_1(2)} \\ \vdots \\ \dfrac{\partial}{\partial w_1(L_w)} \end{bmatrix} \left[w_1(j) r_1(Mk + l - j + 1) + \cdots + w_{lw}(j) r_{Lw}(Mk + l - j + 1) \right] \quad (23)$$

The first partial with respect to w_1 results in only the first term of the sum remaining since it is the only term to depend on w_1. Likewise, for the remaining partials a similar result is obtained.

$$\nabla_{\underline{w_1}} \sum_{j=1}^{Lw} w_1(j) r_1(Mk + l - j + 1) = \begin{bmatrix} \dfrac{\partial}{\partial w_1(1)} w_1(1) r_1(Mk + l - 1 + 1) \\ \dfrac{\partial}{\partial w_1(2)} w_1(2) r_1(Mk + l - 2 + 1) \\ \vdots \\ \dfrac{\partial}{\partial w_1(L_w)} w_1(L_w) r_1(Mk + l - L_w + 1) \end{bmatrix} \quad (24)$$

Now continuing with the partial derivative. Using the first partial as an example,

$$\frac{\partial}{\partial w_1(1)} w_1(1) r_1(Mk + l - 1 + 1) =$$

$$\left(\frac{\partial}{\partial w_{1,R}(1)} + j \frac{\partial}{\partial w_{1,I}(1)} \right) \left[w_{1,R}(1) + j w_{1,R}(1) \right] r_1(Mk + l - 1 + 1) \quad (25)$$

Noting that the cross terms go to zero,

$$\frac{\partial}{\partial w_1(1)} w_1(1) r_1(Mk + l - 1 + 1) = \frac{\partial}{\partial w_{1,R}(1)} w_{1,R}(1) r_1(Mk + l - 1 + 1)$$

$$+ j^2 \frac{\partial}{\partial w_{1,I}(1)} w_{1,R}(1) r_1(Mk + l - 1 + 1) \quad (26)$$

$$\frac{\partial}{\partial w_1(1)} w_1(1) r_1(Mk + l - 1 + 1) = r_1(Mk + l - 1 + 1) + j^2 r_1(Mk + l - 1 + 1) \quad (27)$$

$$\frac{\partial}{\partial w_1(1)} w_1(1) r_1(Mk + l - 1 + 1) = 0 \quad (28)$$

Likewise for the remaining partial derivatives they all go to zero providing the following result,

$$\nabla_{\overline{w_1}} S_{i,out} = 0 \quad (29)$$

Providing the following result for $\nabla_{\overline{w_1}} e_i$,

$$\nabla_{\overline{w_1}} e_i = 0 \quad (30)$$

It is noted that ∇e_i also goes to zero in an identical fashion if the gradient is taken over w_2. Looking at $\nabla_{\overline{w_1}} e_i^*$ in a similar fashion as above for the first TEQ filter w_1,

$$\nabla_{\overline{w_1}} e_i^* = \nabla_{\overline{w_1}} \xi^* - \nabla_{\overline{w_1}} S_{i,out}^* \quad (31)$$

$$\nabla_{\overline{w_1}} e_i^* = -\nabla_{\overline{w_1}} S_{i,out}^* \quad (32)$$

Focusing just on $\nabla_{\overline{w_1}} S_{i,out}^*$,

$$\nabla_{\overline{w_1}} S_{i,out}^* = \nabla_{\overline{w_1}} FEQ_i^* \sum_{\ell=1}^{N} \mathcal{F}_{l,i}^* \sum_{j=1}^{Lw} (w_1^*(j) r_1^*(Mk + l - j) + w_2^*(j) r_2^*(Mk + l - j + 1)) \quad (33)$$

$$\nabla_{\overline{w_1}} S_{i.out}^* = -FEQ_i^* \sum_{\ell=1}^{N} \mathcal{F}_{l,i}^* \nabla_{\overline{w_1}} \sum_{j=1}^{Lw} w_1^*(j) r_1^*(Mk + l - j + 1) \quad (34)$$

Similarly to the $\nabla_{\overline{w_1}} S_{i,out}$ the following results are reached,

$$\nabla_{\overline{w_1}} \sum_{j=1}^{Lw} w_1(j)^* r_1^*(Mk+l-j+1) = \begin{bmatrix} \frac{\partial}{\partial w_1(1)} w_1(1) * r_1 * (Mk+l-1+1) \\ \frac{\partial}{\partial w_1(2)} w_1(2)^* r_1^*(Mk+l-2+1) \\ \vdots \\ \frac{\partial}{\partial w_1(L_w)} w_1(L_w)^* r_1^*(Mk+l-L_w+1) \end{bmatrix} \quad (35)$$

$$\frac{\partial}{\partial w_1(1)} w_1(1)^* r_1^*(Mk+l-1+1) = \frac{\partial}{\partial w_{1,R}(1)} w_{1,R}(1)^* r_1^*(Mk+l-1+1) \quad \cdots$$

$$-j^2 \frac{\partial}{\partial w_{1,I}(1)} w_{1,R}(1)^* r_1^*(Mk+l-1+1) \quad (36)$$

$$\nabla_{\overline{w_1}} \sum_{j=1}^{Lw} w_1(j)^* r_1^*(Mk+l-j+1) =$$

$$\begin{bmatrix} 2r_1^*(Mk+l-1+1) \\ 2r_1^*(Mk+l-2+1) \\ \vdots \\ 2r_1^*(Mk+l-L_w+1) \end{bmatrix} = 2\overline{r_1}^*(Mk+l) \quad (37)$$

$$\nabla_{\overline{w_1}} S_{i,out}{}^* = 2FEQ_i^* \sum_{l=1}^{N} \mathcal{F}_{i,l}^* \overline{r_1}^*(Mk+l) \quad (38)$$

The expression for $\nabla_{\overline{w_1}} e_i^*$ can be written as,

$$\nabla_{\overline{w_1}} e_i^* = -2 \cdot FEQ_i^* \sum_{l=1}^{N} \mathcal{F}_{i,l}^* \overline{r_1}^*(Mk+l) \quad (39)$$

Recalling that $\nabla_{\overline{w_1}} e_i = 0$, the expression for $J_{MCDD,}$

$$\nabla_{\overline{w_1}} J_{MCDD} i = \left(\nabla_{\overline{w_1}} e_i\right) e_i^* + e_i \left(\nabla_{\overline{w_1}} e_i^*\right) \quad (40)$$

$$\nabla_{\overline{w_1}} J_{MCDD} i = e_i \left(\nabla_{\overline{w_1}} e_i^* \right) \tag{41}$$

$$\nabla_{\overline{w_1}} J_{MCDD} i = e_i \left(-2 \cdot FEQ_i^* \sum_{\ell=1}^{N} \mathcal{F}_{i,l}^* \overline{r_1}^* (Mk + l) \right) \tag{42}$$

If similar analysis is done for TEQ w_2, a similar result is found except that received sequence r_2 is used instead of r_1.

$$\nabla_{\overline{w_2}} J_{MCDD} i = e_i \left(-2 \cdot FEQ_i^* \sum_{\ell=1}^{N} \mathcal{F}_{i,l}^* \overline{r_2}^* (Mk + l) \right) \tag{43}$$

By using a gradient descent approach, an update rule for the TEQ w_1 and TEQ w_2 can be found.

Table 1. Multi Carrier Decision Directed Update Rule

$$e_i = \xi\left(S_{i,out}\right) - S_{i,out} \tag{16}$$

$$\bar{r}_{\{1,2\}} = r_{\{1,2\}} (Mk + l + 1 : -1 : Mk + l - L_w + 1)^T \tag{44}$$

$$\nabla_{\overline{w_1}} J_{MCDD} i = e_i \left(-2 \cdot FEQ_i^* \sum_{\ell=1}^{N} \mathcal{F}_{i,l}^* \overline{r_1}^* (Mk + l) \right) \tag{42}$$

$$\nabla_{\overline{w_2}} J_{MCDD} i = e_i \left(-2 \cdot FEQ_i^* \sum_{\ell=1}^{N} \mathcal{F}_{i,l}^* \overline{r_2}^* (Mk + l) \right) \tag{43}$$

$$\overline{w_1}_{k+1} = \overline{w_1}_{k} - \mu \sum_{i=1}^{N} \nabla_{\overline{w_1}} J_{MCDD,i} \tag{45}$$

$$\overline{w_2}_{k+1} = \overline{w_2}_{k} - \mu \sum_{i=1}^{N} \nabla_{\overline{w_2}} J_{MCDD,i} \tag{46}$$

Here equations (16), (42), and (43) are repeated for easy reference and μ is the algorithm step-size.

A good comparison metric between gradient-descent update algorithms is computational complexity. In the case of DD the computational complexity is expected to be low since the error function is first-order. The following summarizes the following operations that must be completed to update the filter.

Table 2. Multi Carrier Decision Directed Computational Complexity

For w_{1k+1}			**Complex Multiplies**	**Adds/Subtracts**
$-\mu\sum_{i=1}^{N}\nabla_{w_1} J_{MCDD,i}$	$\nabla_{w_1} J_{MCDD,i}$			
		Calculating e_i		1 subtract
		e_i	1 complex multiply	
		FEQ_i^*	1 complex multiply	
		$\sum_{l=1}^{N} F_{i,l}^* \bar{r}_1^* (Mk+l+1)$	N complex multiplies x L_w	N adds x L_w
		Repeat N times	x N	x N
	Subtotal $\sum_{i=1}^{N}$		L_w x (N^2+2N)	L_w x (N^2+N) N adds
	μ		1 complex multiply	
	Subtract quantity			L_w Subtracts
Same for w_{2k+1}, assuming $L_{w2} = L_{w1}$			x 2	x 2
Total			2L_w x (N^2+2N)+2	2L_w x (N^2+N+1) +2N

For example, if L_w = 48 and N = 64, it is found that it would require a total of 405506 complex multiplies and 399490 adds/subtracts to update both TEQs. If *N* is

sufficiently large then the N^2 term dominates and the number of Add/Subtract and Multiplies are both of $O\left(N^2\right)$. While the complexity may seem high the results are for N information carrying channels. Thus, the computational complexity per channel is of $O\left(N\right)$ as would be found in the single-carrier case.

Multi Carrier Trained (MCT)

Multi Carrier Trained algorithm is so similar to the MCDD algorithm that it doesn't warrant a new derivation but rather only the differences between MCDD and MCT need to be shown. The only major difference between the MCT and the MCDD algorithm is the error function, e_i. The MCT error function is based upon what the actual transmitted symbol was rather than a decision direction function. It is possible to change (16) of the MCDD derivation to (46) and proceed with virtually no substantial changes.

$$e_i = S_{i,in} - S_{i,out} \tag{47}$$

When (19) and (31) are revisited the following changes are made,

$$\nabla_{\underline{w_1}} e_i = \nabla_{\underline{w_1}} S_{i,in} - \nabla_{\underline{w_1}} S_{i,out} \tag{48}$$

$$\nabla_{\underline{w_1}} e_i^* = \nabla_{\underline{w_1}} S_{i,in}^* - \nabla_{\underline{w_1}} S_{i,out}^* \tag{49}$$

Because the gradient over $S_{i,in}$ does not depend on w_1 or w_2 then the resulting partial derivative with respect to w_1 and w_2 both result in zero leaving the remaining portion of the derivation for MCDD unchanged. When updating w_1 or w_2 be sure to use the MCT error function (46) which lead to the following result,

Table 3. Multi Carrier Trained Update Rule

$$e_i = S_{i,in} - S_{i,out} \tag{47}$$

$$\bar{r}_{\{1,2\}} = r_{\{1,2\}}(Mk + l + 1 : -1 : Mk + l - L_w + 1)^T \tag{50}$$

$$\nabla_{\overline{w_1}} J_{MCT} i = e_i \left(-2 \cdot FEQ_i^* \sum_{l=1}^{N} \mathcal{F}_{i,l}^* \overline{r_1}^* (Mk + l + 1) \right) \tag{51}$$

$$\nabla_{\overline{w_2}} J_{MCT} i = e_i \left(-2 \cdot FEQ_i^* \sum_{l=1}^{N} \mathcal{F}_{i,l}^* \overline{r_2}^* (Mk + l + 1) \right) \tag{52}$$

$$\overline{w_{1\,k+1}} = \overline{w_{1\,k}} - \mu \sum_{i=1}^{N} \nabla_{\overline{w_1}} J_{MCT,i} \tag{53}$$

$$\overline{w_{2\,k+1}} = \overline{w_{2\,k}} - \mu \sum_{i=1}^{N} \nabla_{\overline{w_2}} J_{MCT,i} \tag{54}$$

The changes outlined above show that both MCT and DD have very similar derivation and hence the computational cost for MCDD is the same as it would be for MCT (see Table 2). Because of the differences in the error functions, MCT will have better performance than MCDD as well as not have the same convergence problems even under high noise or large channel effects.

Multi Carrier Constant Modulus (MCCM)

The MCCM algorithm formulates an error based on a second-order difference between the magnitude squared of the input symbol, the modulus, and the magnitude squared of the output symbol. If *i* designates the output of interest then the MCCM cost function can be written,

$$J_{MCCM} = \sum_{i=1}^{N} J_{MCCM,i} = \sum_{i=1}^{N} e_i e_i^* \tag{55}$$

Where e_i is the error and e_i^* is error conjugate

$$e_i = 2 - \left|S_{i,out}\right|^2 = 2 - \left(S_{i,out}\right)\left(S_{i,out}{}^*\right) \qquad (56)$$

Since the modulus represents the square of the distance from the origin on the complex plane, the error function e_i, is in effect the smallest distance from the circle that circumscribes the magnitude squared of an input symbol, see Figure 5 for illustration of 4-QAM modulus and Figure 6 for visual representation of the error function.

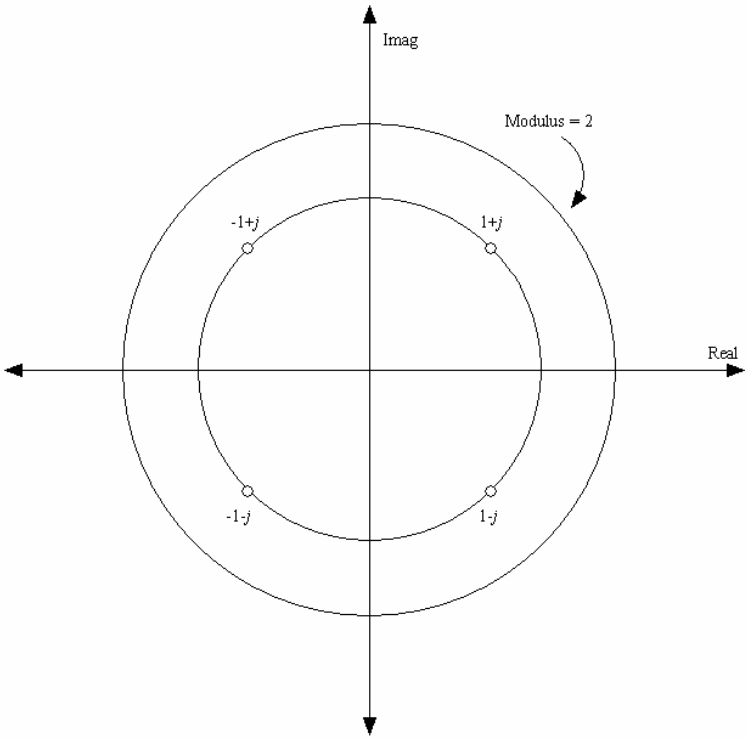

Figure 5. Illustration of 4-QAM Modulus

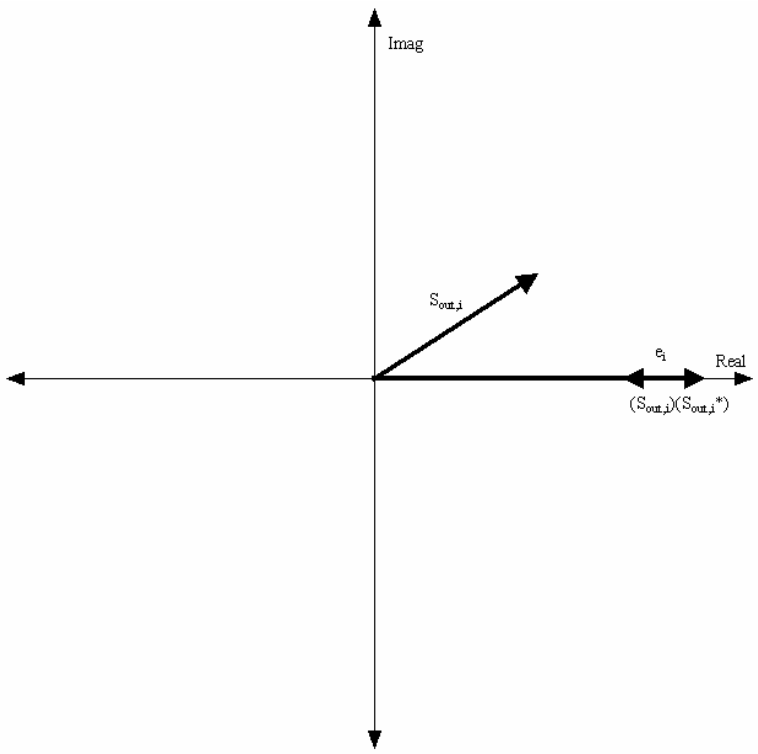

Figure 6. Visual Representation of Constant Modulus (CM) Error, e_i

In Figure 6, e_i is the vector on the real axis that points left. The error, e_i, does not capture any error attributed with symbol rotation and only serves to reduce the error in magnitude. Also of note is that the error function is completely real. This second-order error function leads to a fourth-order cost function. As described in the section on the DD update algorithm, a higher-order cost function means increased computation and slower convergence when near the minimum solution. In tradeoff, it is found that CM has better convergence farther away from the minimum solution and when the BER of the system is high, since the modulus of the system is the same for all input symbols. For

many blind, single-carrier systems, CM is often used on cold start up, and when the BER falls low enough, a switch is made to DD.

There will be a corresponding error for each of the tones in the MC communication system thus the goal is to reduce the sum of the cost functions for each tone. By taking the gradient with respect to the TEQ filter on each tone an update rule is derived that ensures that the cost function moves towards a minimum.

$$\nabla_{\underline{w_1}} J_{MCCM,i} = \nabla_{\underline{w_1}} \left(e_i e_i^* \right) \tag{57}$$

$$\nabla_{\underline{w_1}} J_{MCCM,i} = \left(\nabla_{\underline{w_1}} e_i \right) e_i^* + e_i \left(\nabla_{\underline{w_1}} e_i^* \right) \tag{58}$$

Looking just at $\nabla_{\underline{w_1}} e_i$ for the first TEQ filter w_1,

$$\nabla_{\underline{w_1}} e_i = \nabla_{\underline{w_1}} 2 - \nabla_{\underline{w_1}} \left(S_{i,out} \right) \left(S_{i,out}{}^* \right) \tag{59}$$

Since 2 is a constant then its gradient with respect to the TEQ w_1 is always zero.

$$\nabla_{\underline{w_1}} e_i = -\left[\left(\nabla_{\underline{w_1}} S_{i,out} \right) \left(S_{i,out}{}^* \right) + \left(S_{i,out} \right) \left(\nabla_{\underline{w_1}} S_{i,out}{}^* \right) \right] \tag{60}$$

Focusing on $\nabla_{\underline{w_1}} S_{i,out}$, Recall $S_{i,out}$,

$$S_{i,out} = FEQ_i \sum_{l=1}^{N} \mathcal{F}_{i,l} \sum_{j=1}^{Lw} ((w_1(j) r_1(Mk + l - j + 1) + w_2(j) r_2(Mk + l - j + 1) \tag{61}$$

The term $w_2(j) r_2(Mk + l - j + 1)$ is ignored since it will resolve to zero when the gradient is taken with respect to $\overline{w_1}$.

$$\nabla_{\overline{w_1}} S_{i,out} = \nabla_{\overline{w_1}} FEQ_i \sum_{l=1}^{N} \mathcal{F}_{i,l} \sum_{j=1}^{Lw} w_1(j) r_1(Mk + l - j + 1) \qquad (62)$$

The gradient can be moved inside the FEQ and Fourier coefficient since they do not depend on $\overline{w_1}$,

$$\nabla_{\overline{w_1}} S_{i,out} = FEQ_i \sum_{l=1}^{N} \mathcal{F}_{i,l} \nabla_{\overline{w_1}} \sum_{j=1}^{Lw} w_1(j) r_1(Mk + l - j + 1) \qquad (63)$$

When moving the gradient inside the sum a little more work has to be done to progress further. Examining the sum more closely,

$$\nabla_{\overline{w_1}} \sum_{j=1}^{Lw} w_1(j) r_1(Mk + l - j + 1) =$$

$$\begin{bmatrix} \frac{\partial}{\partial w_1(1)} \\ \frac{\partial}{\partial w_1(2)} \\ \vdots \\ \frac{\partial}{\partial w_1(L_w)} \end{bmatrix} \left[w_1(j) r_1(Mk + l - j + 1) + \cdots + w_{lw}(j) r_{Lw}(Mk + l - j + 1) \right] \qquad (64)$$

The first partial with respect to w_1 results in only the first term of the sum remaining since it is the only term to depend on w_1. Likewise for the remaining partials a similar result occurs.

$$\nabla_{\underline{w_1}} \sum_{j=1}^{Lw} w_1(j) r_1(Mk + l - j + 1) = \begin{bmatrix} \frac{\partial}{\partial w_1(1)} w_1(1) r_1(Mk + l - 1 + 1) \\ \frac{\partial}{\partial w_1(2)} w_1(2) r_1(Mk + l - 2 + 1) \\ \vdots \\ \frac{\partial}{\partial w_1(L_w)} w_1(L_w) r_1(Mk + l - L_w + 1) \end{bmatrix} \quad (65)$$

Continuing with the partial derivative. Using the first partial as an example,

$$\frac{\partial}{\partial w_1(1)} w_1(1) r_1(Mk + l - 1 + 1) =$$

$$\left(\frac{\partial}{\partial w_{1,R}(1)} + j \frac{\partial}{\partial w_{1,I}(1)} \right) \left(w_{1,R}(1) + j w_{1,R}(1) \right) r_1(Mk + l - 1 + 1) \quad (66)$$

Noting that the cross terms go to zero,

$$\frac{\partial}{\partial w_1(1)} w_1(1) r_1(Mk + l - 1 + 1) = \frac{\partial}{\partial w_{1,R}(1)} w_{1,R}(1) r_1(Mk + l - 1 + 1)$$

$$+ j^2 \frac{\partial}{\partial w_{1,I}(1)} w_{1,R}(1) r_1(Mk + l - 1 + 1) \quad (67)$$

$$\frac{\partial}{\partial w_1(1)} w_1(1) r_1(Mk + l - 1 + 1) = r_1(Mk + l - 1 + 1) + j^2 r_1(Mk + l - 1 + 1) \quad (68)$$

$$\frac{\partial}{\partial w_1(1)} w_1(1) r_1(Mk + l - 1 + 1) = 0 \quad (69)$$

Likewise for the remaining partial derivatives they all go to zero providing the following result,

$$\nabla_{\underline{w_1}} S_{i,out} = 0 \quad (70)$$

Focusing just on $j \nabla_{\underline{w_1}} S_{i,out}^{*}$ whose derivation is very similar to $\nabla_{\underline{w_1}} S_{i,out}$,

$$\nabla_{\overline{w_1}} S_{i,out}{}^{*} = \nabla_{\overline{w_1}} FEQ_i^{*} \sum_{\ell=1}^{N} \mathcal{F}_{i,l}^{*} \sum_{j=1}^{Lw} w_1(j)^{*} r_1^{*}(Mk+l-j+1) \tag{71}$$

$$\nabla_{\overline{w_1}} S_{i,out}{}^{*} = FEQ_i^{*} \sum_{\ell=1}^{N} \mathcal{F}_{i,l}^{*} \nabla_{\overline{w_1}} \sum_{j=1}^{Lw} w_1(j)^{*} r_1^{*}(Mk+l-j+1) \tag{72}$$

$$\nabla_{\overline{w_1}} \sum_{j=1}^{Lw} w_1(j)^{*} r_1^{*}(Mk+l-j+1) = \begin{bmatrix} \frac{\partial}{\partial w_1(1)} w_1(1) * r_1 * (Mk+l-1+1) \\ \frac{\partial}{\partial w_1(2)} w_1(2)^{*} r_1^{*}(Mk+l-2+1) \\ \vdots \\ \frac{\partial}{\partial w_1(L_w)} w_1(L_w)^{*} r_1^{*}(Mk+l-L_w+1) \end{bmatrix} \tag{73}$$

$$\frac{\partial}{\partial w_1(1)} w_1(1)^{*} r_1^{*}(Mk+l-1+1) = \frac{\partial}{\partial w_{1,R}(1)} w_{1,R}(1)^{*} r_1^{*}(Mk+l-1+1)$$

$$- j^2 \frac{\partial}{\partial w_{1,I}(1)} w_{1,R}(1)^{*} r_1^{*}(Mk+l-1+1) \tag{74}$$

$$\nabla_{\overline{w_1}} \sum_{j=1}^{Lw} w_1(j)^{*} r_1^{*}(Mk+l-j+1) =$$

$$\begin{bmatrix} 2r_1^{*}(Mk+l-1+1) \\ 2r_1^{*}(Mk+l-2+1) \\ \vdots \\ 2r_1^{*}(Mk+l-L_w+1) \end{bmatrix} = 2\overline{r_1}^{*}(Mk+l) \tag{75}$$

$$\nabla_{\overline{w_1}} S_{i,out}{}^{*} = 2FEQ_i^{*} \sum_{\ell=1}^{N} \mathcal{F}_{i,l}^{*} \overline{r_1}^{*}(Mk+l) \tag{76}$$

Going back to (60) and substituting in (70) and (86),

$$\nabla_{\overline{w_1}} e_i = -\left(S_{i,out}\right)\left(2FEQ_i^{*} \sum_{\ell=1}^{N} \mathcal{F}_{i,l}^{*} \overline{r_1}^{*}(Mk+l)\right) \tag{77}$$

Now looking at $\nabla_{\overline{w_1}} e_i^{*}$ since e_i is all-real error signal then by virtue $e_i^{*} = e_i$ thus,

$$\nabla_{\overline{w}_1} e_i^{\,*} = -\left(S_{i,out}\right)\left(2FEQ_i^* \sum_{l=1}^{N} \mathcal{F}_{i,l}^{\,*}\, \overline{r}_1^{\,*}(Mk+l)\right) \tag{78}$$

Revisiting (66),

$$\nabla_{\overline{w}_1} J_{MCCM,i} = \left(\nabla_{\overline{w}_1} e_i\right) e_i^* + e_i \left(\nabla_{\overline{w}_1} e_i^*\right) \tag{79}$$

$$\nabla_{\overline{w}_1} J_{MCCM,i} = -2 \cdot S_{i,out} \cdot FEQ_i^* \sum_{l=1}^{N} \mathcal{F}_{i,l}^{\,*}\, \overline{r}_1^{\,*}(Mk+l+1)\left(e_i^* + e_i\right) \tag{80}$$

$$\nabla_{\overline{w}_1} J_{MCCM,i} = -4 \cdot e_i \cdot S_{i,out} \cdot FEQ_i^* \sum_{l=1}^{N} \mathcal{F}_{i,l}^{\,*}\, \overline{r}_1^{\,*}(Mk+l) \tag{81}$$

A similar result would be found if the gradient is taken with respect to w_2,

$$\nabla_{\overline{w}_2} J_{MCCM,i} = -4 \cdot e_i \cdot S_{i,out} \cdot FEQ_i^* \sum_{l=1}^{N} \mathcal{F}_{i,l}^{\,*}\, \overline{r}_2^{\,*}(Mk+l) \tag{82}$$

By using a gradient descent approach, update rules for the TEQs w_1 and w_2 are found.

Table 4. Multi Carrier Constant Modulus Update Rule

$$e_i = 2 - \left(S_{i,out}\right)^2 = 2 - \left(S_{i,out}\right)\left(S_{i,out}^{\,*}\right) \tag{56}$$

$$\overline{r}_{\{1,2\}} = r_{\{1,2\}}(Mk+l+1:-1:Mk+l-L_w+1)^T \tag{83}$$

$$\nabla_{\overline{w}_1} J_{MCCM,i} = -4 \cdot e_i \cdot S_{i,out} \cdot FEQ_i^* \sum_{l=1}^{N} \mathcal{F}_{i,l}^{\,*}\, \overline{r}_1^{\,*}(Mk+l) \tag{81}$$

$$\nabla_{\overline{w}_2} J_{MCCM,i} = -4 \cdot e_i \cdot S_{i,out} \cdot FEQ_i^* \sum_{l=1}^{N} \mathcal{F}_{i,l}^{\,*}\, \overline{r}_2^{\,*}(Mk+l) \tag{82}$$

$$\overline{w}_{1\,k+1} = \overline{w}_{1\,k} - \mu \sum_{i=1}^{N} \nabla_{\overline{w}_1} J_{MCCM,i} \tag{84}$$

$$\overline{w}_{2\,k+1} = \overline{w}_{2\,k} - \mu \sum_{i=1}^{N} \nabla_{\overline{w}_2} J_{MCCM,i} \tag{85}$$

Here equations (81) and (82) are repeated for easy lookup and μ is the algorithm step-size.

Examining the computational complexity similarly to DD, the following results are found for CM. For CM it is expected that computational complexity would be higher since the error function is second-order. The following summarizes the following operations that must be completed to update the filter.

Table 5. Multi Carrier Constant Modulus Computational Complexity

For w_{1k+1}			**Complex Multiplies**	**Adds/Subtracts**
$-\mu\sum_{i=1}^{N}\nabla_{w_1}J_{MCCM,i}$	$\nabla_{w_1}J_{MCCM,i}$			
		Calculating e_i	1 complex multiply	1 subtract
		e_i	1 complex multiply	
		$S_{i,out}$	1 complex multiply	
		FEQ_i^*	1 complex multiply	
		$\sum_{l=1}^{N} F_{i,l}^* r_1^{-*}(Mk+l+1)$	N complex multiplies x L_w	N adds x L_w
		Repeat N times	x N	x N
	Subtotal $\sum_{i=1}^{N}$		L_w x (N^2+4N)	L_w x (N^2+N) N adds
	μ		1 complex multiply	
	Subtract quantity			L_w Subtracts
Same for w_{2k+1}, assuming $L_{w2} = L_{w1}$			x 2	x 2
Total			$2L_w$ x (N^2+4N)+2	$2L_w$ x (N^2+N+1) +2N

It is quickly noted that the number of adds/subtracts as well as the number of multiplies has increased above what is required for DD thus confirming that the trend for CM to have higher computational complexity than DD in the single-carrier case applies when modified for MC. For example, If $L_w = 48$ and $N = 64$, it requires a total of 417794 complex multiplies and 399490 adds/subtracts to update both TEQs which requires an additional 12288 complex multiplies, a negligible increase over MCDD, the number of adds/subtracts remain the same. When comparing to single carrier update algorithms it should be recognized that this is for *N*-separate channels and that the computational complexity per channel is a better metric to compare.

Least Mean Squared Frequency Domain Equalizers

Frequency Domain Equalizers (FEQs) are important in helping to resize the symbol constellation and to remove rotation in the symbol set. It is often the case that TEQs require re-normalization, such as a unit norm constraint, after each update to keep them from growing without bound, keeping them from being able to correct a change in size of the symbol constellation. Because the TEQ needs to re-normalize it falls on the FEQ to resize the symbol constellation. FEQs are not always required to make confident bit-error determinations as is the case when differential encoding is used but they do help open the eye diagram and decrease output symbol error. Similar to TEQs, FEQs can be trained or blind depending on the information available at the receiver. Like trained TEQs, trained FEQs have faster convergence and have the ability to de-rotate the symbol set even if it has rotated an arbitrary number of quarter turns. Blind FEQs must rely on

information about the transmitted symbol set in order to de-rotate the symbol set resulting in slower convergence than the trained case and the inability to correctly remove a phase rotation of more than a quarter turn. The two FEQs implemented and derived in this section consist of one that is developed when training information is available, LMS-FEQ, and one based on a Decision Direction (DD) cost function, DDLMS-FEQ. These FEQs are adaptive and use a Least Mean Squares approach to finding the direction of lowest cost. The differences seen between DD and trained are minimal as seen in the difference between MCDD and MCT. The following will be a discussion of DDLMS-FEQ and to follow will be an explanation of how to derive a trained LMS-FEQ through minor changes.

Starting with a cost function for the single-tap FEQ, for output *i*, the cost function should be the magnitude squared of the error function,

$$J_{FEQ_i} = e_i e_i^* \tag{86}$$

Where e is the error and e* is error compliment

$$e_i = \xi\left(S_{i,out}\right) - S_{i,out} \tag{87}$$

Here ξ is the decision direction function whose output is the closest symbol in the communication system's constellation. By taking the gradient with respect to the FEQ filter ensures that the cost functions moves towards a minimum.

$$\nabla_{FEQ_i} J_{FEQ_i} = \nabla_{FEQ_i}\left(e_i e_i^*\right) \tag{88}$$

$$\nabla_{FEQ_i} J_{FEQ_i} = \left(\nabla_{FEQ_i} e_i\right) e_i^* + e_i\left(\nabla_{FEQ_i} e_i^*\right) \tag{89}$$

Looking just at $\nabla_{FEQ_i} e_i$,

$$\nabla_{FEQ_i} e_i = \nabla_{FEQ_i} \xi - \nabla_{FEQ_i} S_{i,out} \tag{90}$$

Since ξ always resolves to a constant then the gradient of ξ with respect to the FEQ is always zero. Recall $S_{i,out}$,

$$S_{i,out} = FEQ_i \sum_{l=1}^{N} \mathcal{F}_{i,l} \sum_{j=1}^{Lw} (w_1(j) r_1(Mk + l - j + 1) + w_2(j) r_2(Mk + l - j + 1)) \tag{91}$$

Equation (91) can further be simplified by rewriting the formula in terms of the output after the FFT, H_i , because only the first part depends on the FEQ,

$$S_{i,out} = FEQ_i H_i \tag{92}$$

Now examining $\nabla_{FEQ_i} S_{i,out}$,

$$\nabla_{FEQ_i} S_{i,out} = \nabla_{FEQ_i} FEQ_i H_i \tag{93}$$

The only part that depends on the FEQ is of interest when taking the partial derivative.

Examining $\nabla_{FEQ_i} FEQ_i$ more closely**,**

$$\nabla_{FEQ_i} FEQ_i = \left(\nabla_{FEQ_{i,R}} + j\nabla_{FEQ_{i,I}}\right)\left(FEQ_{i,R} + jFEQ_{i,I}\right) \tag{94}$$

Noting that the cross terms go to zero,

$$\nabla_{FEQ_i} FEQ_i = \nabla_{FEQ_{i,R}} FEQ_{i,R} + j^2 \nabla_{FEQ_{i,I}} FEQ_{i,I} \tag{95}$$

Completing the gradients,

$$\nabla_{FEQ_i} FEQ_i = 1 + j^2 = 0 \tag{96}$$

Thus,

$$\nabla_{FEQ_i} S^*_{i,out} = 0 \tag{97}$$

Now moving on to the $\nabla_{FEQ_i} e^*_i$,

$$\nabla_{FEQ_i} e^*_i = \nabla_{FEQ_i} \xi^* - \nabla_{FEQ_i} S^*_{i,out} \tag{98}$$

Since ξ^* always resolves to a constant then the gradient of ξ with respect to the FEQ is always zero. Recall $S^*_{i,out}$,

$$S^*_{i,out} = FEQ^*_i \sum_{l=1}^{N} \mathcal{F}^*_{i,l} \sum_{j=1}^{Lw} (w_1^*(j) r_1^*(Mk + l - j + 1) + w_2^*(j) r_2^*(Mk + l - j + 1)) \tag{99}$$

Equation (99), like equation (91), can further be simplified because only the first part depends on the FEQ,

$$S^*_{i,out} = FEQ^*_i H^*_i \tag{100}$$

Now examining $\nabla_{FEQ_i} S^*_{i,out}$,

$$\nabla_{FEQ_i} S^*_{i,out} = \nabla_{FEQ_i} FEQ^*_i H^*_i \tag{101}$$

The only part that depends on the FEQ is of interest when taking the partial derivative.

Examining $\nabla_{FEQ_i} FEQ_i$ more closely,

$$\nabla_{FEQ_i} FEQ^*_i = \left(\nabla_{FEQ_{i,R}} + j\nabla_{FEQ_{i,I}}\right)\left(FEQ_{i,R} - jFEQ_{i,I}\right) \tag{102}$$

Noting that the cross terms go to zero,

$$\nabla_{FEQ_i} FEQ^*_i = \nabla_{FEQ_{i,R}} FEQ_{i,R} - j^2 \nabla_{FEQ_{i,I}} FEQ_{i,I} \tag{103}$$

$$\nabla_{FEQ_i} FEQ^*_i = 1 - j^2 = 2 \tag{104}$$

Thus,

$$\nabla_{FEQ_i} S^*_{i,out} = 2H^*_i \tag{105}$$

Revisiting equations (89) and substituting equations (90), (97), (98), and (105),

$$\nabla_{FEQ_i} J_{FEQ_i} = \left(\nabla_{FEQ_i} e_i\right) e^*_i + e_i \left(\nabla_{FEQ_i} e^*_i\right) \tag{89}$$

$$\nabla_{FEQ_i} J_{FEQ_i} = e_i \left(-\nabla_{FEQ_i} S^*_{i,out}\right) \tag{106}$$

$$\nabla_{FEQ_i} J_{FEQ_i} = -2e_i H^*_i \tag{107}$$

Rewriting some equations for ease of searching, Table 6 puts together the adaptive update rule for a blind LMS-FEQ.

Table 6. Blind LMS-FEQ Update Rule

$$e_i = \xi\left(S_{i,out}\right) - S_{i,out} \tag{87}$$

$$H_i = \sum_{l=1}^{N} \mathcal{F}_{i,l} \sum_{j=1}^{Lw} (w_1(j) r_1(Mk + l - j + 1) + w_2(j) r_2(Mk + l - j + 1)) \tag{108}$$

$$\nabla_{FEQ_i} J_{FEQ_i} = -2e_i H^*_i \tag{107}$$

$$FEQ_i = FEQ_i - \mu \nabla_{FEQ_i} J_{FEQ_i} \tag{108}$$

In Table 6, μ is the algorithm step size. Table 7 examines the computational complexity of the blind LMS-FEQ and tabularizes the results. The computational complexity is minor in comparison to updating the TEQ because most of the information required is already provided as output of the communication system before the FEQ.

Table 7. LMS FEQ Computational Complexity

For FEQ_i			**Complex Multiplies**	**Adds/Subtracts**
$-\mu\nabla_{FEQ_i} J_{FEQ_i}$	$\nabla_{FEQ_i} J_{FEQ_i}$			
		Calculating e_i		1 subtract
		e_i	1 complex multiply	
		H_i^*	1 complex multiply	
	μ		1 complex multiply	
	Subtract quantity			1 subtract
		Repeat N times	x N	x N
Total			3N	2N

The cost for a single FEQ update is repeated N times because there it is repeated for every output channel. Taking into account our previous example of N = 64, the cost for updating the multicarrier FEQ every update requires 256 complex multiples and 128 adds/subtracts.

Next, the derivation for a trained FEQ will be shown from derivation for the blind FEQ. By making a simple change to the error function of the blind FEQ, a trained FEQ is derived with minimal effort. Consider the following change to the error function,

$$e_i = S_{i,in} - S_{i,out} \tag{109}$$

Since $S_{i,in}$ always resolves to a constant then the gradient of $S_{i,in}$ with respect to the FEQ is always zero providing a similar case as in the blind FEQ. Because both cases are similar the analysis follows the same logic all the way to the conclusion. See Table 7 for a trained adaptive update rule for a LMS-FEQ.

Table 8. Trained LMS-FEQ Update Rule

$$e_i = S_{i,in} - S_{i,out} \quad (109)$$

$$H_i = \sum_{\ell=1}^{N} \mathcal{F}_{i,l} \sum_{j=1}^{Lw} (w_1(j) r_1(Mk + l - j + 1) + w_2(j) r_2(Mk + l - j + 1)) \quad (110)$$

$$\nabla_{FEQ_i} J_{FEQ_i} = -2 e_i H_i^* \quad (111)$$

$$FEQ_i = FEQ_i - \mu \nabla_{FEQ_i} J_{FEQ_i} \quad (112)$$

Examining the computational cost for adaptively updating the FEQ using a training sequence and LMS approach it is found that it has nearly the same cost as a blind FEQ. Table 9 shows the computational cost of a trained LMS-FEQ.

Table 9. LMS FEQ Computational Complexity

For FEQ_i			**Complex Multiplies**	**Adds/Subtracts**
$-\mu \nabla_{FEQ_i} J_{FEQ_i}$	$\nabla_{FEQ_i} J_{FEQ_i}$			
		Calculating e_i		1 subtract
		e_i	1 complex multiply	
		H_i^*	1 complex multiply	
	μ		1 complex multiply	
	Subtract quantity			1 subtract
		Repeat N times	x N	x N
Total			3N	2N

Again, the cost for a single FEQ update is repeated N times because there it is repeated for every output channel. Taking into account our previous example of N = 64, the cost for updating the multicarrier FEQ every update requires 192 complex multiples and 128 adds/subtracts.

Summary

This chapter has examined several blind and trained algorithms for adapting both Time-domain Equalizers (TEQs) as well as Frequency-domain Equalizers (FEQs). Multi Carrier Decision Directed (MCDD), a blind adaptive equalization algorithm based upon the single carrier version of decision directed (DD) was developed and the update rule provided. In addition, the computational complexity for MCDD was evaluated. Next, Multi Carrier Trained (MCT) was derived as a small change to MCDD and its computational complexity was noted as comparable to MCDD. Following MCDD and MCT was the derivation for Multi Carrier Constant Modulus (MCCM), a blind adaptive equalization algorithm based upon the single carrier version of constant modulus (CM), as well as results for its computational complexity. Finally, the chapter finished with the derivation for both blind and trained frequency domain equalizers (FEQs) and an examination of their computational complexity.

V. Communication System Operation

Overview

This section describes the many common pieces used in setting up the communication system and how parameter selection is achieved. All of the algorithms are run using the system model presented in Chapter 2 and are set up with similar parameters to allow comparison among the algorithms. The rest of this section focuses on describing the system model and general parameter selection while the remaining sections of this chapter discuss more involved parameters.

The transmitter and channel portion of the system model (IFFT, P/S&CP, h_1 and h_2, n_1 and n_2) has been implemented with an Inverse Fast Fourier Transform (IFFT) size of 64, a Cyclic Prefix (CP) of 16, and an h_1 and h_2 length of 32. The receiver portion of the system model (w_1 and w_2, S/P&XCP, FFT, FEQ) is implemented with w_1 and w_2 having a length of 48 as well as other values corresponding to those in the transmitter such as the XCP having a removal length 16 and the FFT having a size of 64, while the FEQ consists of single tap filters.

Default initialization of w_1 and w_2 is achieved by setting a center tap of the filters w_1 and w_2 to a constant value of one. In all simulations default initialization is done by setting tap 24 to a value of one. The value "one" is used because the filters w_1 and w_2 are renormalized to have constant energy equal to one and initializing the energy of the filter to anything other one would just result in immediate renormalization to one. In addition,

tap 24 is chosen because it is nearly an equal distance from either end of the filter. An equalization filter will have its best performance if the majority of the energy and ripple is in the center of the filter rather than on the edges due to the fact that taps below zero and above 48 are effectively all zeros and abrupt changes cause poor performance. Default initialization of the FEQ is done by setting all FEQ taps to 1. Setting the FEQ taps to 1 initially introduces no rotation or scaling to the output sequence.

Channel Selection

A Rayleigh distributed fading channel with an exponential decay profile was used to implement the channel, see Sklar [24] for implementation of Rayleigh channels. The following MATLAB code snippet shows how the Rayleigh fading channel was created,

```
Fade1 = 0.1;
ChannelLength1 = 32;

PowerScalingFactor1 = exp(-Fade1*(0:ChannelLength1-1));

RayleighChannel1 = PowerScalingFactor1.*(randn(1,ChannelLength1)
             +j*randn(1,ChannelLength1));
```

Note that the channel length has been chosen longer than the CP. If the channel length is not longer than the CP then all channel interference can be removed by the CP and the TEQ doesn't need to do any adaption to shorten the channel. In all examples, a channel length of 32 is used with a Rayleigh channel that has a fade value of 0.1, providing a channel that is twice as long as our CP ensuring that enough energy will reside outside the window of 16 samples that eye diagram closure will result if the channel is not

shortened, even with a high signal to noise ratio (SNR). The following figure is a magnitude plot of an example Rayleigh channel,

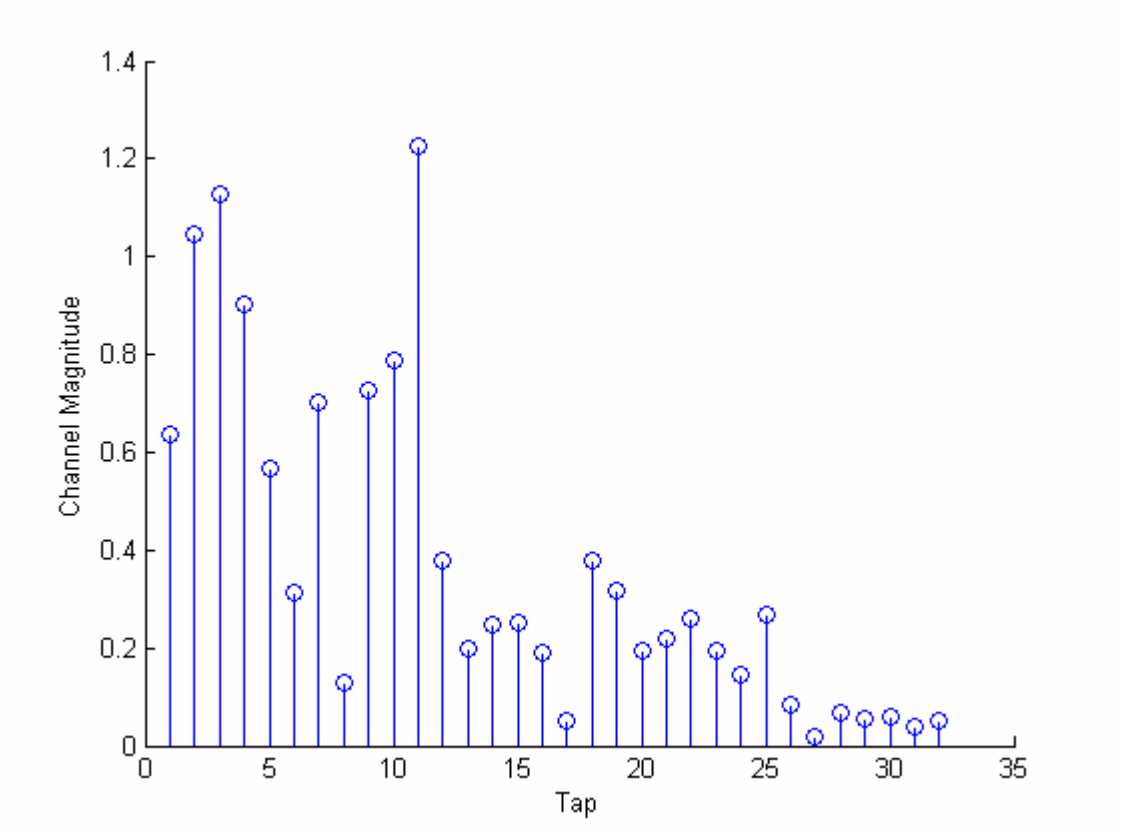

Figure 7. Magnitude Plot of Rayleigh Fading Channel

Noise Generation

After introducing our Rayleigh fading channel noise is added. While not all noise is white and Gaussian, for the purposes of the simulations, white Gaussian noise provides a good basis for evaluating the algorithms against each other. Other noise sources as well as narrow-band interferers could be examined to see how these algorithms perform under non-standard conditions. The complex noise sequence is generated by selecting a desired SNR value in decibels (SNRDB) and scaling a complex white Gaussian sequence

according to the computed SNR. The SNR is the ratio of average signal power to average noise power. If a 4-QAM constellation is used for input symbols to the FFT then each input symbol has a signal power of σ_x^2.

$$\sigma_x^2 = 2 \tag{113}$$

Since there are 64 carriers, when the average is taken over all the carriers the signal power remains the same as for a single carrier.

$$\sum_{i=1}^{64} \frac{\sigma_x^2}{64} = 2 \tag{114}$$

The noise power, σ_n, is then scaled appropriately, by the term *NoisePower* to achieve a desired SNR.

$$SNR = \frac{\left(norm(h_1)\right)^2 \sum_{i=1}^{64} \frac{\sigma_x^2}{64}}{\left(NoisePower\right)^2 \sigma_n^2} \tag{115}$$

Since σ_x^2 is defined to have a power of 2, similar to σ_x^2. To obtain a specific SNR value, solve to determine the appropriate scaling factor for the noise power.

$$NoisePower = \frac{norm(h_1)}{\sqrt{SNR}} \tag{116}$$

In the case where null carriers are present, the average signal power decreases since some of the carriers have zero noise power. For example, when there are 12 null carriers present the following SNR is developed.

$$SNR = \frac{\left(norm(h_1)\right)^2 \sum_{i=1}^{52} \frac{\sigma_x^2}{64}}{\left(NoisePower\right)^2 \sigma_n^2} \tag{117}$$

The noise power is then adjusted to achieve the desired SNR.

$$NoisePower = \frac{norm(h_1)\sqrt{\frac{52}{64}}}{\sqrt{SNR}} \qquad (118)$$

See the following MATLAB code snippet for development of the noise sequence for when there are no null carriers.

```
SNR = 10^(SNRDB/10);
NoisePower =norm(Channel1)*sqrt(1/SNR);

NoisePower*(randn(1,length(ChannelOut1)) +
      j*randn(1,length(ChannelOut1)));
```

For implementing the Fractionally Spaced Equalizer (FSE) a second channel is generated as well as a second noise sequence using the same process.

Selecting Step Size

Step size plays a critical role in ensuring that the algorithms move toward a minimum and will depend largely upon the channel and noise power. Selecting a step size that is too large can cause the algorithm to diverge. An example of how to select a good step size will be shown using MCT and then step sizes for all of the algorithms will be given using that method.

The algorithm adaption step size μ needs to be selected for both TEQs as well as the FEQ. Appropriate step sizes for the MCT algorithm are shown in the following MATLAB code snippet,

```
W1delta = .000003;
W2delta = .000003;
FEQdelta = .005;
```

These values were chosen through experimentation by selecting the values that provided the fastest convergence of the MCT algorithm.

The first step in obtaining good algorithm step sizes is to ensure that the FEQ adapts before the TEQ since the TEQ will try to shorten the channel to an impulse if the FEQ is too slow. In order to find the best FEQ step size, set the step size for the TEQs, w_1 and $w_{2,}$ to zero (not adapting) and adjust the FEQ step size until the best performance is achieved. FEQ performance can be measured by observing the FEQ error function, or sum of error functions, over the adaption window and choosing the step size that provides the fastest convergence with the smallest error. A larger step size usually results in a larger error but faster convergence; step sizes too large cause the algorithm to diverge. After the FEQ step size is chosen, the next step is to choose an appropriate step size for the TEQ. Experimentation will help to choose a good step size for the TEQs. Repeatedly examine the cost function over the window of updates for a range of step sizes and choose the step size that provides the fastest convergence.

The process for selecting the step size for all of the adaptive algorithms is essentially the same. Test and select a good step size for the FEQ then move on to selecting a good step size for the TEQ. The following table shows the step sizes found to be sufficient using this test and select method,

Table 10. Algorithm Adaption Step Sizes

	FEQ	TEQ
Multicarrier Decision Directed	0.005	0.000003
Multicarrier Constant Modulus	0.005	0.00000005
Multicarrier Trained	0.005	0.000003
Multicarrier Equalization by Restoration of Redundancy	0.005	0.0005
Carrier Nulling	0.005	0.000003

Null Carriers

The authors presenting CNA show that often OFDM systems use the first K and last K channels of the OFDM system as guard channels to ensure that the system isn't encroached upon. These null carriers can be used to blindly equalize the channel. See De Courville et al. [17] as well as Romano and Barbarossa [18], for full explanation of how CNA is implemented. When null carriers are referenced in this context of this research it is implemented as the first 6 and last 6 carriers of the OFDM system transmitting a constant string of zeros. Because the adaptive equalization algorithms behave differently in the presence of null carriers, analysis of the algorithms is treated along two distinct classes, with or without null carriers. The primary driving factor for the special treatment for null carriers is that the MERRY algorithm assumes that the input sequences are uncorrelated and the addition of null carriers creates correlation, see Martin et al.[8].

All algorithms when run with null carriers are treated with the knowledge that they know that the first 6 and last 6 carriers are null carriers and when training or adapting they either use that to their advantage such as MCT and CNA or ignore the information such as MCDD and MCCM. MERRY is unchanged since its cost function is derived from the time-domain samples. Null carriers are not seen as an advantage for MERRY because they introduce correlation into the time-domain samples. CNA, since it relies solely on null carriers is not examined when comparing algorithms without null carriers.

Synchronization Delay

Synchronization delay plays a key role in determining how well the TEQ can shorten the channel and will be the first metric of comparison for the algorithms. It is important to understand how synchronization delay affects the algorithms performance in order to get the best performance possible. Since the largest $v+1$ taps of the channel are usually the first $v+1$ taps in the model used in this thesis, the channel delay is zero. Because the channel doesn't introduce a delay, the ideal synchronization delay is one less than that introduced by setting tap 24 equal to 1. If the channel were to introduce a delay the ideal synchronization delay would be the delay introduced by the channel plus the delay introduced by the equalization filter.

The BER curves generated using the Rayleigh fading channel and default initialization of the TEQs have the synchronization delay value set to 23 since it is a near optimal value across all the algorithms. With the introduction of null carriers the synchronization delay will need to be reevaluated to ensure that 23 is a good synchronization delay value. It is assumed that the algorithms will perform similarly with respect to synchronization delay when null carriers are present as when they are not present.

Multi Carrier Trained (MCT) Example

MCT is a great example algorithm because it establishes how well you can do with perfect transmitted symbol knowledge and demonstrates the effects of an adaptive shortening algorithm. Although the other algorithms go about shortening the channel in different ways according to different cost functions they all strive to do the same thing

which can be clearly illustrated using MCT. After MCT has had plenty of time to shorten the channel, comparisons will be made between equalized and un-equalized plots.

The scatter diagram in Figure 8 for output data generated using a synchronization delay of 23, SNR equal to 15 dB, and no TEQ or FEQ adaptive algorithms running shows that the eye is closed. The eye diagram is closed mostly due to channel effects since the noise doesn't contribute substantially to eye closure as a SNR of 15 dB is considered a fairly good operating noise level.

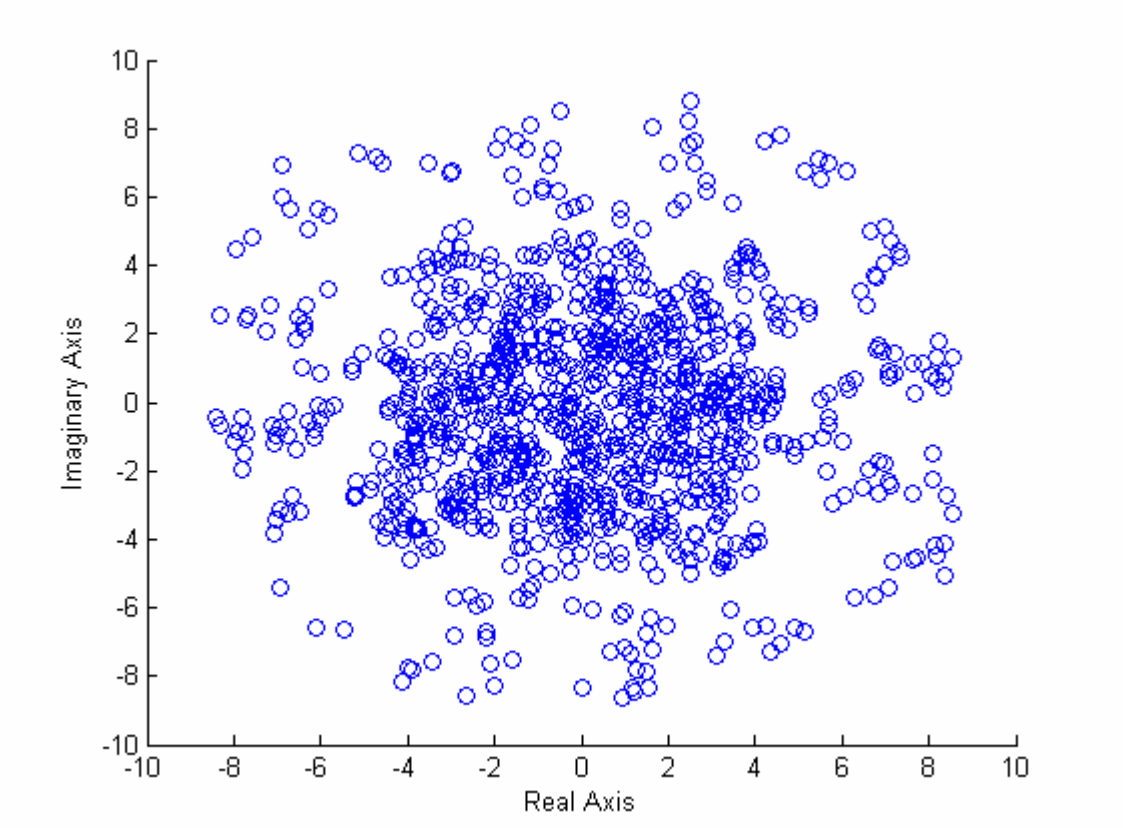

Figure 8. Plot Showing Unequalzied Scatter Diagram

Along with eye diagram analysis, another good plot to examine is the effective channel since it shows in which taps the equalized and un-equalized energies reside. The

effective channel is the TEQ convolved with the Rayleigh channel. In the case of the FSE, the effective channel is the sum of both TEQs convolved with their respective Rayleigh channels. The following MATLAB code snippet shows how the effective channel is generated for a FSE,

```
EffectiveChannel = (conv(RayleighChannel1,w1)+conv(RayleighChannel2,w2));
```

If the Rayleigh channel has length 32 and w_1 and w_2 have length 48 then the effective channel will have a length of 32+48 -1, or equivalently 79 taps. Figure 9 shows what the real part of the effective channel looks like before equalization using default parameters.

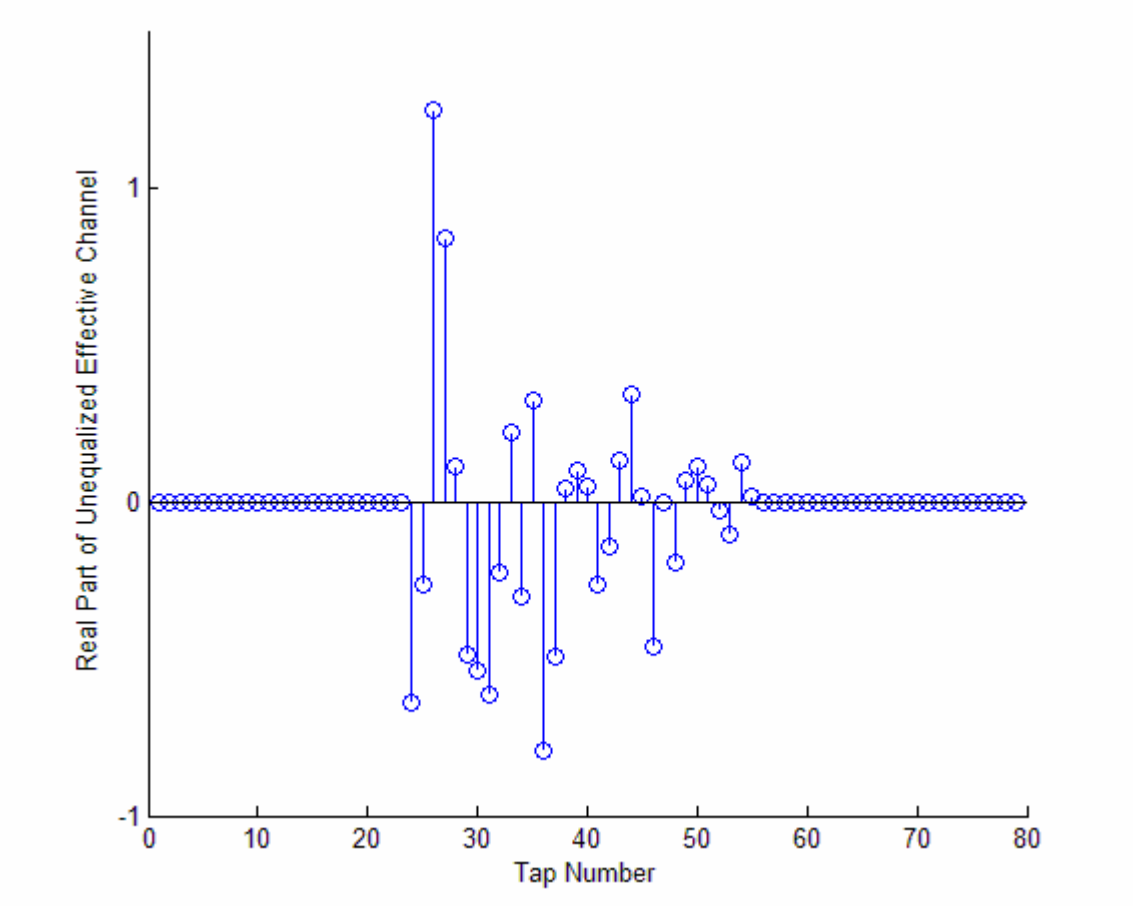

Figure 9. Plot Showing Unequalized Effective Channel

Note that the un-equalized effective channel has energy in more than 17 taps. The CP in the default example can only remove the distortion attributable to 17 taps of the effective channel. For systems that do not have a TEQ the only option available is to increase the CP to handle a longer effective channel increasing overhead. Later it will be shown that a TEQ can reorient the energy outside of the first 16 taps and move it so that the effective channel is shorter.

When the MCT algorithm is run for 10,000 updates using a step size of 0.000003 for the TEQ and a step size of 0.005 for the FEQ, results for the effective channel as well

as a scatter diagram showing an open eye are obtained. Relative to Figure 9, Figure 10 illustrates the shortening of the effective channel, i.e. there are fewer non-zero tap values.

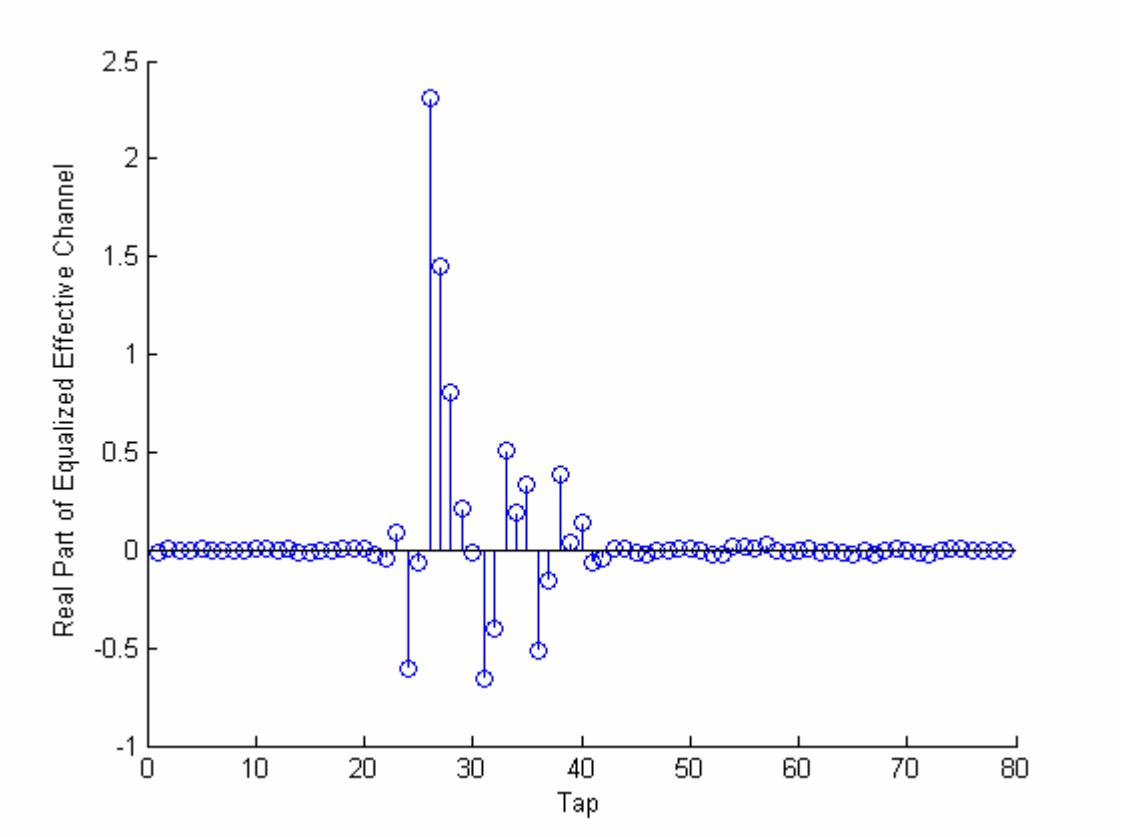

Figure 10. MCT Equalized Effective Channel

As can be seen in the above shortened channel, the TEQ shortening filter has moved the energy residing outside of the range correctable by the CP. Also, Figure 10 shows that there are 17 taps over which a large majority of the channel energy resides. Since the CP can remove interference due to 16 taps the 17th tap with energy in it is seen as just a delay. A comparison plot showing the real part of the energy that was moved is detailed in Figure 11.

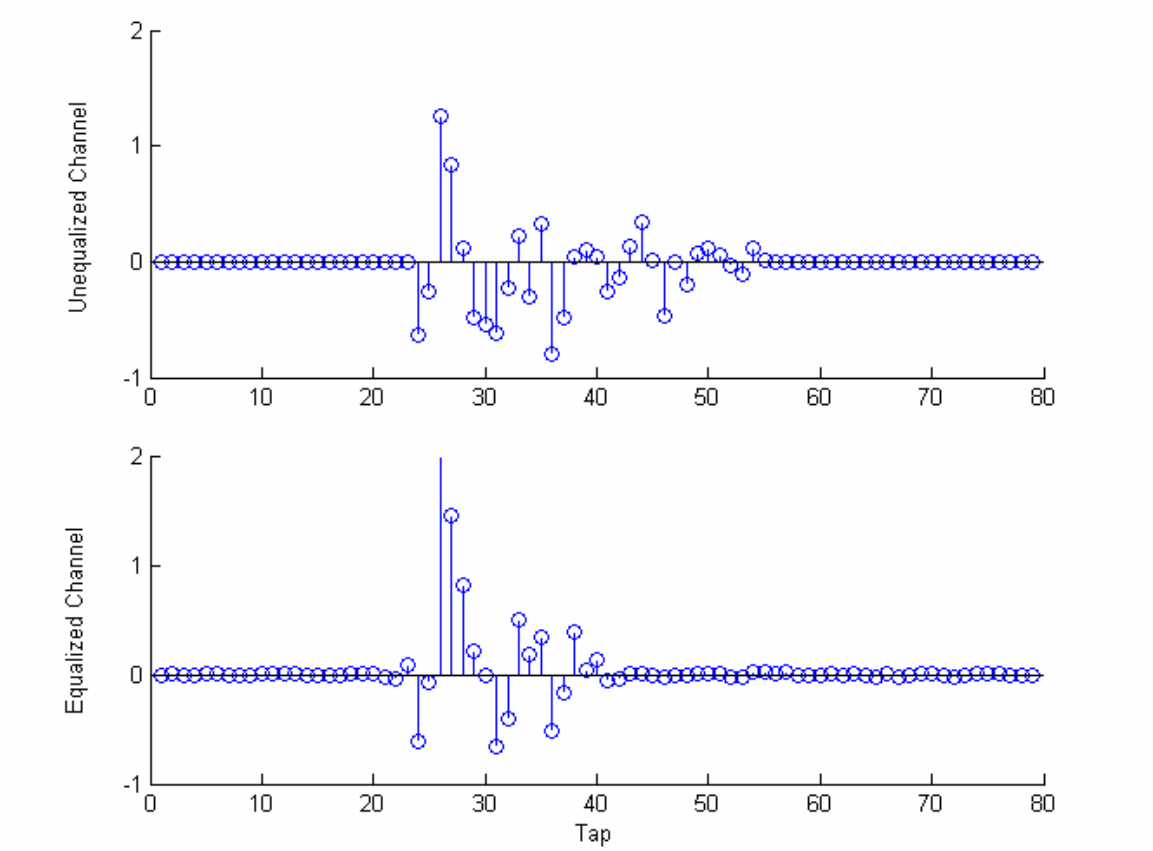

Figure 11. Comparison Plot Between MCT Equalized and Unequalized Effective Channels

After the channel has been successfully shortened, the plot in Figure 12 of the output scatter diagram is seen to have removed the effects of ISI and ICI, causing the scatter diagram eye to open. Figure 12 shows the eye diagram after MCT has shortened the channel.

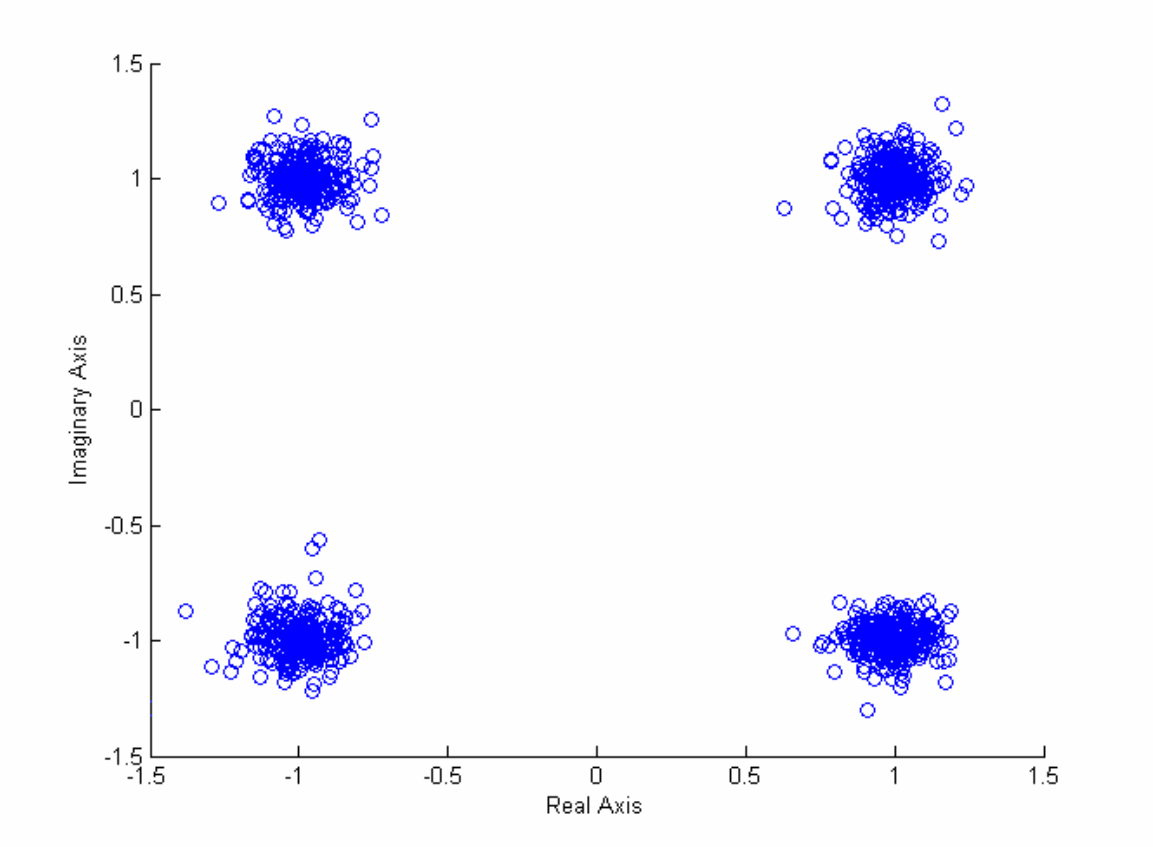

Figure 12. MCT Equalized Scatter Diagram

Figure 13 shows that the MCT adaption algorithm is reducing the cost function, equivalently the mean squared error (MSE).

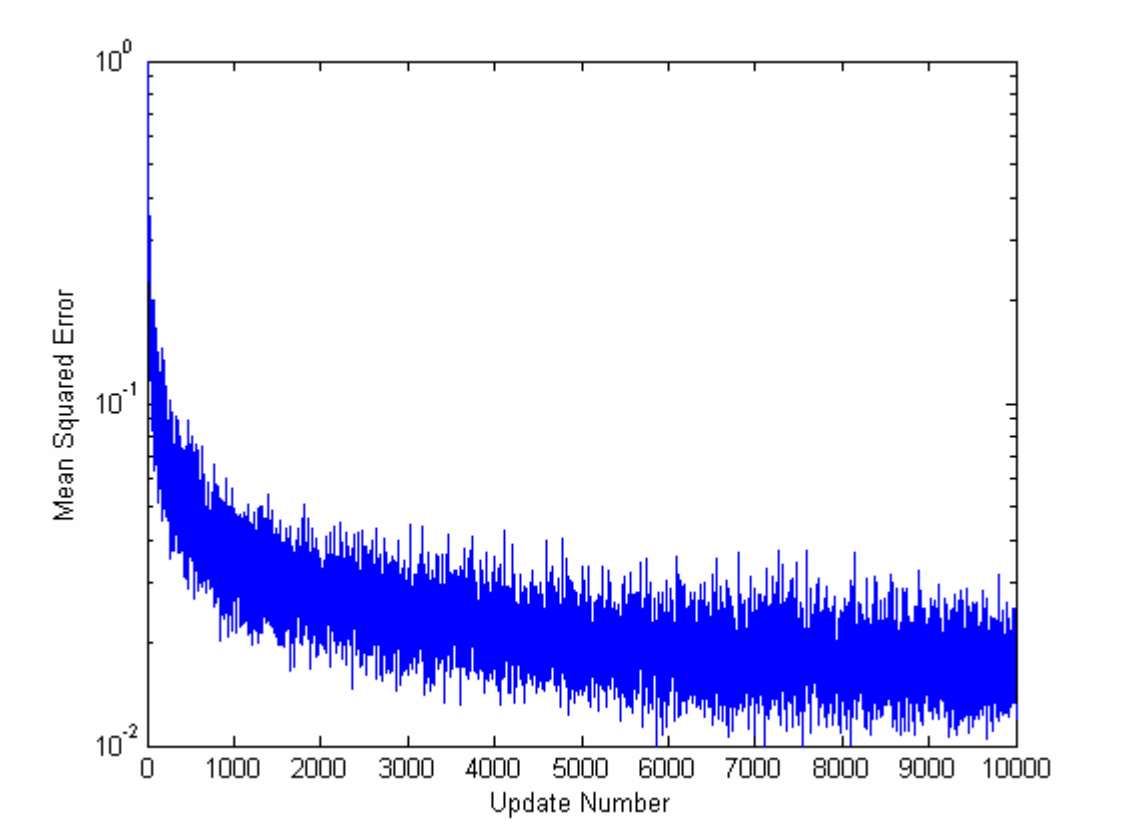

Figure 13. MCT Cost Function

As was noted earlier, MCT has mostly adapted to the channel after 3000 updates. Further updates after 3000 have not provided a substantial decrease in the error. Most of the sharp drop and bend seen between updates 0 and 1000 are attributed to the FEQ adapting which causes a resizing and de-rotation of the output symbol constellation.

All of the plots shown in this section have provided conclusive evidence that the MCT algorithm has shortened the channel as well as opened the eye diagram. The remaining algorithms are blind and will not do as well as MCT at shortening the channel. The blind algorithms strive to provide the same results as the MCT so replicating the plots shown for MCT is not necessary for the rest of the algorithms.

VI. Results and Analysis

Overview

This chapter focuses on exercising and comparing Multi Carrier Trained (MCT), Multi Carrier Decision Directed (MCDD), and Multi Carrier Constant Modulus (MCCM) with each other and with two additional blind adaptive equalization algorithms, Multicarrier Equalization by Restoration of Redundancy (MERRY) and Carrier Nulling Algorithm (CNA). The goal of examining the new algorithms with each other, as well as several well-accepted blind adaptive equalization algorithms, is to stratify the algorithms according to BER in order to get a feel for the proper way to implement the algorithms. This chapter provides comparisons of all algorithms showing bit error (BER) vs. SNR curves for each of the algorithms as well as BER vs. synchronization delay plots, both in the presence of null carriers and without.

Synchronization Delay without Null Carriers

Synchronization delay plays a key role in determining the performance of the algorithm and it would be an advantage to determine how a change in synchronization delay affects BER. In the set up of the communication system there is a set delay introduced by the initialization of the TEQ and a delay in the channel. The optimal synchronization delay is always one less than the sum of the two delays. In order to obtain the plots in this section each algorithm shortened the same channel for a range of synchronization delay values and the BER was observed after each algorithm had

performed 1000 updates. The following are plots showing BER versus synchronization delay for the four algorithms simulated without null carriers,

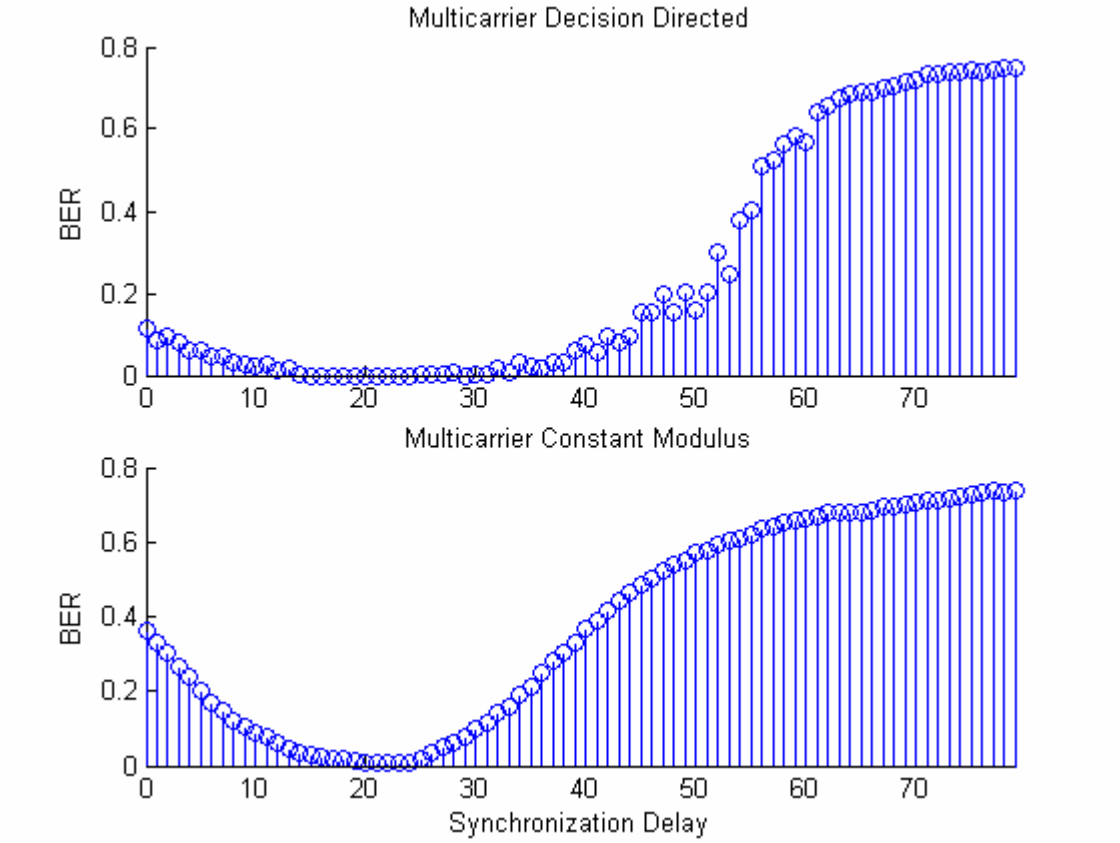

Figure 14. Plot of BER vs. Synchronization Delay for MCDD and MCCM without Null Carriers

The above plot shows that 23 is indeed an optimal synchronization delay for MCDD and MCCM as well as showing that there is a window of good values that can be chosen. Because the algorithms have only had 1000 updates there are correspondingly fairly high BER rates, but the low number of updates was chosen to clearly show that synchronization delay has substantial effect on BER and even convergence rate. The next figure shows MERRY and MCT without null carriers present,

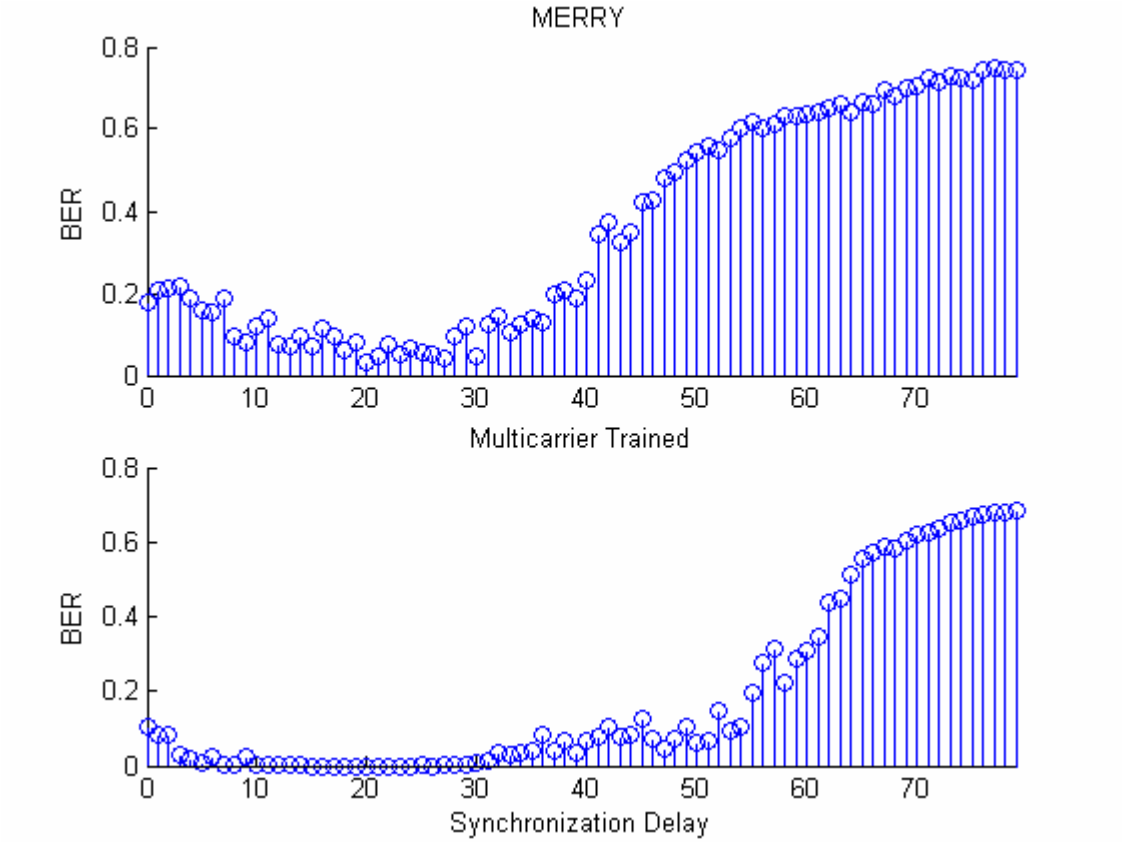

Figure 15. Plot of BER vs. Synchronization Delay for MERRY and MCT without Null Carriers

It can be seen from both Figure 13 and Figure 14 that all algorithms have the same area of lowest BER around a synchronization delay of 23. As noted earlier, MCT can be seen to have the best performance of all the algorithms. Since MCT has perfect transmitted symbol knowledge it has a larger area of good synchronization delay values. The next section will examine synchronization delay for the case when null-carriers are present.

Synchronization Delay with Null Carriers

After examining the system without null carriers and determining the range of values that provide good convergence and consequentially low BERs, focus can move on

to the case when null carriers are present in the transmitter. In this section the CNA algorithm will also be shown since with the availability of null carriers brings the ability to use CNA.

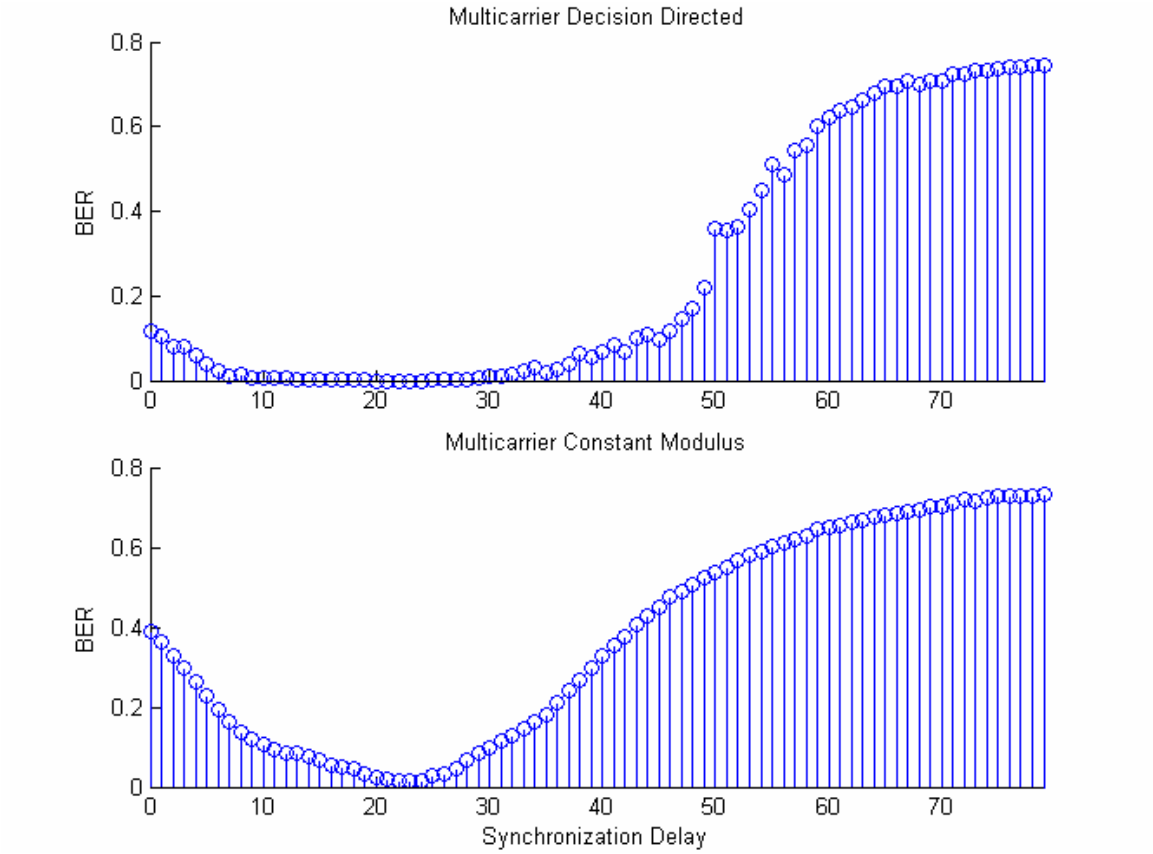

Figure 16. Plot of BER vs. Synchronization Delay for MCDD and MCCM with Null Carriers

The following plot shows MERRY and MCT when null carriers are present,

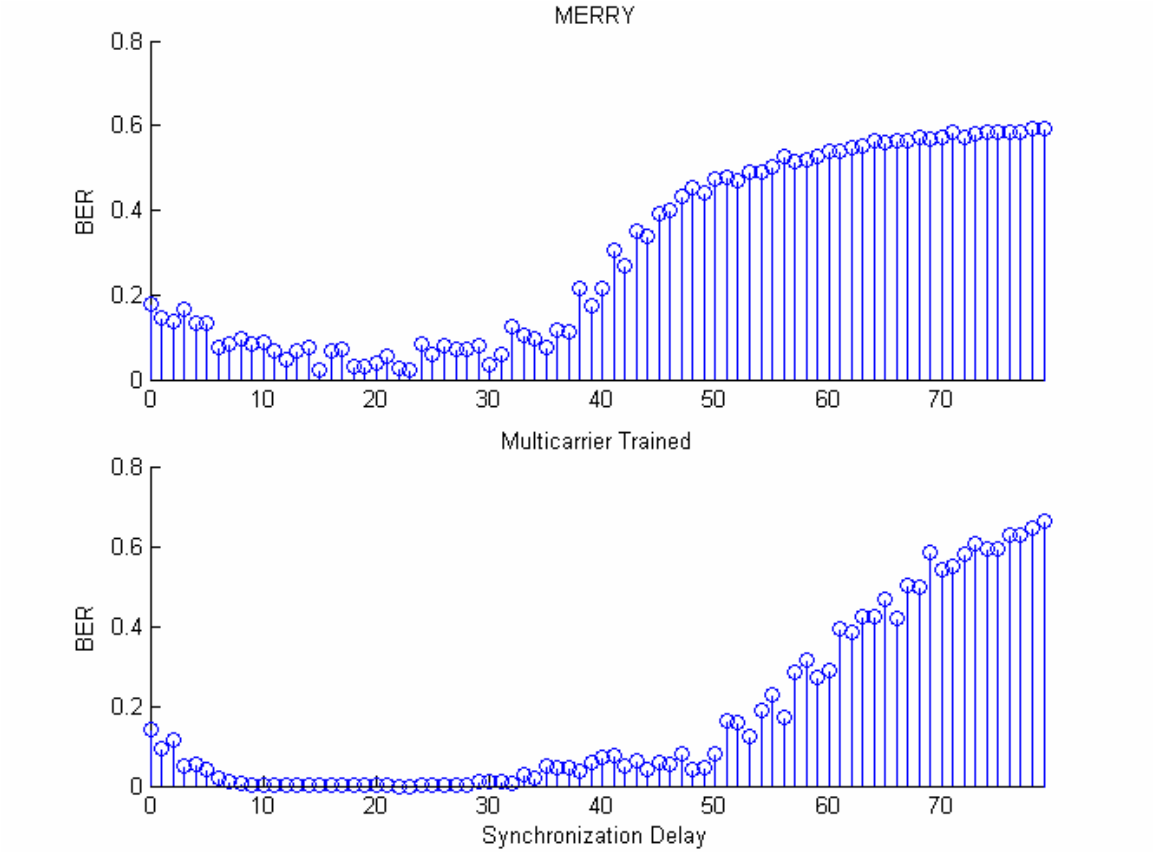

Figure 17. Plot of BER vs. Synchronization Delay for MERRY and MCT with Null Carriers

The following plot shows CNA when null carriers are present,

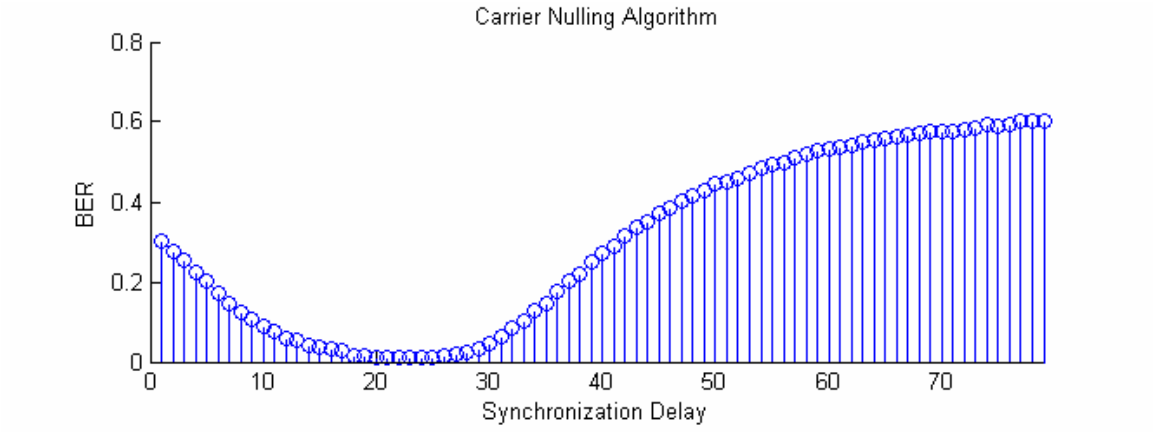

Figure 18. Plot of BER vs. Synchronization Delay for CNA with Null Carriers

Again Figure 16, Figure 17 and Figure 18 show that the addition of null carriers doesn't have a substantial affect on changing the selection of a good synchronization delay as well as present new results for CNA. A selection of a delay around 23 provides again provides good results for ensuring fast convergence and low BERs. As stated before in the previous chapter, the ideal synchronization delay will always be the delay introduced by the TEQ plus the delay introduced by the channel.

The results for the previous two sections on synchronization delay are drawn over the same channels with a fairly good SNR of 15 dB. A more comprehensive study of synchronization delay vs. BER could be done where the algorithms were allowed to adapt over a larger number of updates and for a wide variety of Rayleigh fading channels so that more general conclusions can be drawn about good synchronization delay values.

Bit Error Rate vs. Signal to Noise Ratio without Null Carriers

A plot of BER vs. SNR reveals an expected performance given an operating SNR. In addition, plotting the results for all the algorithms gives a chance the viewer a chance to compare results and stratify algorithm performance. In this section the simulations focus on the case when null carriers are not present and so results for CNA will not be shown. The next section shows results for when null carriers are present.

The following plot shows simulation results for MCT, MCDD, MCCM, and MERRY,

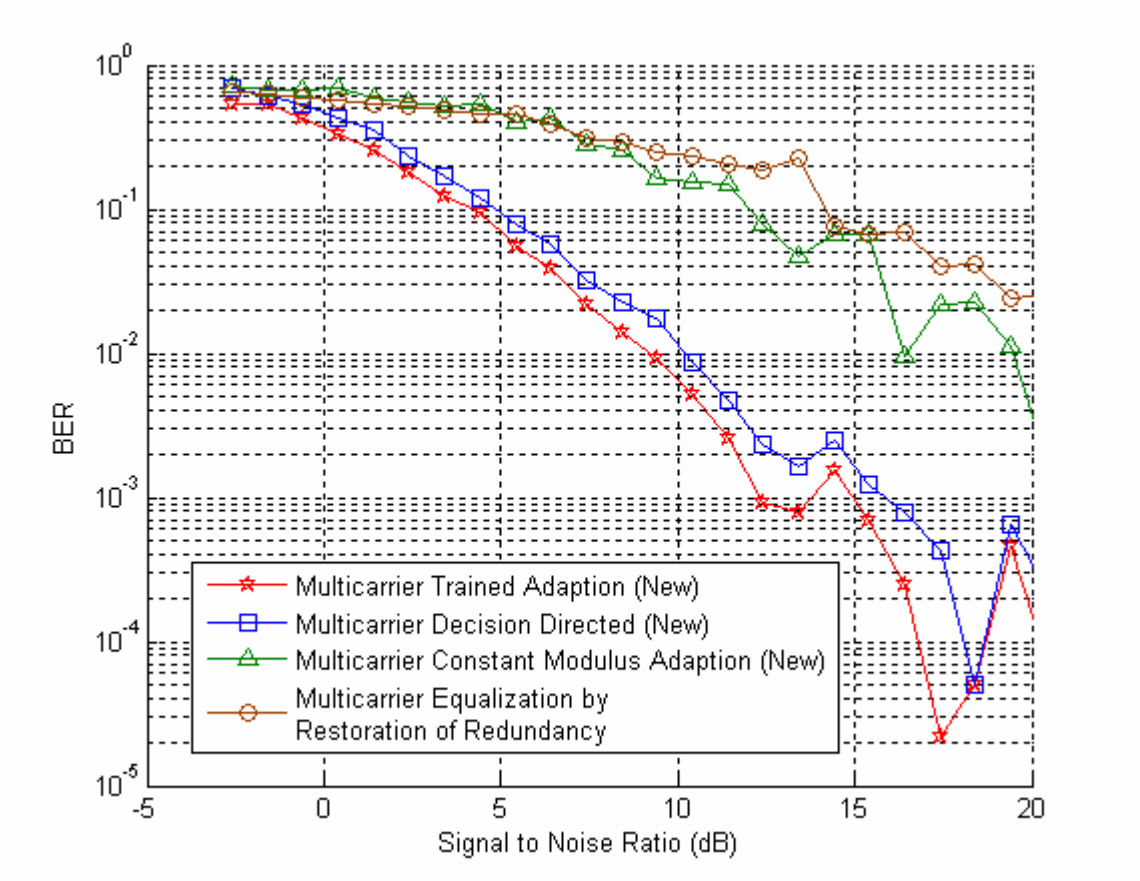

Figure 19. BER vs. SNRdB without Null Carriers

Since MCT uses perfect transmitted symbol information it has the best performance followed closely by MCDD. MCDD has performance very close to MCT since at very high SNR the decision direction function provides output nearly exact to the input sequence. A second grouping of BER curves comes with MCCM, and MERRY. Even though MERRY only performs slightly worse than MCCM it should be noted that MERRY has a slow convergence rate requiring more updates than the other blind adaptive algorithms. MERRY would benefit substantially from an additional 5,000-10,000 updates and would provide an improved BER curve. Even though MERRY requires additional updates to fully converge, MERRY requires much less computation

than the other algorithms both trained and blind, a benefit when processing speed is important.

Results in this section have shown BERs after 10,000 updates of each respective algorithm. MCT and MCDD, which have nearly identical performance for high SNR, show better channel equalization than MERRY and MCCM. MERRY is showing a worse performance than MCCM but since MERRY converges slowly it could benefit from additional updates. The next section will focus on detailing BER vs. SNR dB with null carriers present.

Bit Error Rate vs. Signal to Noise Ratio with Null Carriers

Null carriers introduce correlation into the transmitted data which may produce a shift in BER for some of the adaptive algorithms. Since MCT knows that certain data channels will transmit a zero its performance isn't expected to change. MCDD and MCCM cost functions depend on knowing information about the transmitted symbols and may perform slightly worse since they will have to ignore the 12 null carriers, leaving less channels to adapt with. MERRY is derived based upon having uncorrelated data and because the null carriers introduce correlation it should perform worse.

The following plot shows BER vs. SNR for MCT, MCDD, MCCM, MERRY and CNA,

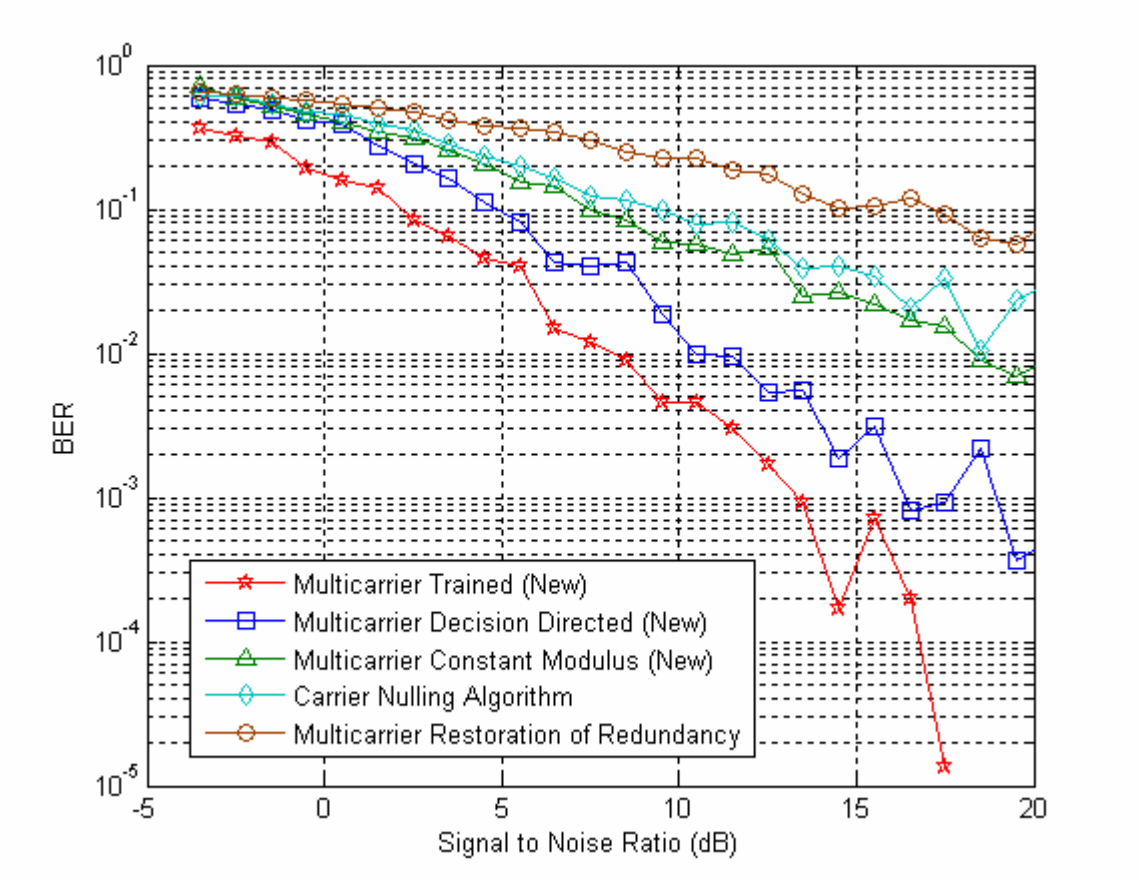

Figure 20. BER vs. SNRdB with Null Carriers

MCT has not changed from the introduction of null carriers and MCDD has not shown any substantial drop in performance. MCCM is showing a BER that is slightly better than CNA but seems to have a slightly worse performance after the null carriers have been introduced.

Results in this section have shown BERs after 10,000 updates of each respective algorithm in the presence of null carriers. Nearly identical to the case when null carriers weren't present, MCT and MCDD have nearly identical performance for high SNR and show better channel equalization than MERRY, MCCM, and CNA. Also as in the non-null carrier case, additional updates will still benefit MERRY.

This chapter showed BER vs. Synchronization delay as well as BER vs. SNR dB for MCT, MCDD, and MCCM with and without null carriers and noted how the algorithms performed in relation with each other as well as several other MC blind adaptive algorithms, MERRY and CNA.

VII. Conclusion

Summary

This section concludes the thesis on blind adaptive equalization algorithms for multicarrier systems by examining the focus and results of the thesis. The focus of this thesis was to develop and exercise two new blind adaptive equalization algorithms, Multi Carrier Decision Directed (MCDD) and Multi Carrier Constant Modulus (MCCM) for adapting a Time Domain Equalizer (TEQ) as well as present new results on a trained multicarrier adaptive equalization algorithm, Multi Carrier Trained (MCT) and a single tap Frequency Domain Equalizer (FEQ).

The reader was introduced to a system model reminiscent of many OFDM systems and notation describing certain operations within the system was outlined. Next, background literature was discussed to give the reader a treatment of current methods for channel shortening as well as example algorithms. Afterward, analytical development of MCDD, MCT, MCCM, LMS-FEQ, and the DDLMS-FEQ was given, and then exploratory results for MCT were shown. The Results and Analysis chapter exercised all the algorithms and presented results showing optimal synchronization delays and BER curves for a range of SNR values.

The primary goal of this research, algorithm derivation and development, was completed and shown explicitly in Chapter 4 and MATLAB implementations of all algorithms are given in Appendix A. The new algorithms were shown to shorten the channel and decrease inter-symbol interference as well as inter-carrier interference,

showing the effectiveness of the new algorithms and meeting the primary research goal. The secondary goal of this research was to compare the newly derived algorithms to other multicarrier adaptive algorithms. MCT, MCDD, and MCCM were simulated and compared against MERRY and CNA in Chapter 6, meeting the secondary research goal.

Additional Research Work

MCT, MCDD, and MCCM have several areas where additional research would benefit the algorithms by characterizing performance under a larger variety of simulations such as pink or colored noise rather than white, the introduction of narrow band interferers, more comprehensive synchronization delay study, and comparison against a wider range of blind TEQ adaptive algorithms.

Noise sequences are often not white and so a study of the algorithms presented for other noise types could provide increased knowledge about the robustness or shortcomings of these algorithms in the presence of colored noise and interferers.

As previously noted, a more comprehensive study of how synchronization delay affects the algorithms should be done with a random sampling of Rayleigh fading channels as well as allowing the algorithms more time to adapt to the channel. With an increased number of updates many of the algorithms will shorten the channel enough so that BERs will be too low at an SNR of 15 dB. To compensate, the algorithms could be run at a lower SNR to introduce more errors so that a more varied plot could be obtained.

Along with a more comprehensive study of how synchronization delay affects the algorithms more multicarrier algorithms, such as Sum Squared Autocorrelation Minimization (SAM) and others, could be compared. The additional comparisons would

give better insight into how the algorithms stack up against other multicarrier blind adaptive equalization algorithms.

Appendix A.

Overview

Appendix A includes MATLAB code segments for implementing the transmitters, channels, and receivers detailed in this thesis. The system model is broken into 2 logical pieces based upon ease of operation for running simulations and aggregating results. First the transmitter and channel portion of the code has been written so that a function call to a transmitter / channel, given a set of parameters, returns appropriate information for running a second function for reception / decoding of the transmitted sequence. See the appropriate functions for arguments passed and returned. Adaptation of the functions for providing additional information is possible depending on simulation need. In order to run the transmitter and receiver functions a script file is set up to make repeated calls to both of these functions. The format of the rest of this appendix will show the script file followed by the transmitter and channel, and lastly show the 5 receiver files.

Script File

The script file is used to make repeated and coordinated calls to the transmitter and channel as well as the appropriate receiver functions. The script file shown is set up to make multiple runs and collect Bit Error Rates (BERs) for all of the algorithms present.

```
% This is a script file to control the transmitter and reciever in order to
% coordinate multiple runs with slightly differnt operating parameters.

clear;
clc;
```

```
tic;

%//////// Default Parameters for Operating the Transmitter   ///////////
DiffEncoding = 1;     % This Turns DiffEncoding on = 1/off = 0
NullCarriers = 0;     % This Turns Null Carriers on = 1/off = 0
FFTSIZE = 64;         % The size of the IFFT/FFT being performed
NUMFFT = 10001;       % The number of FFTs being done
CP = 16;              % Cyclic Prefix
SNRDB = 100;          % Constant to scale Noise Power

ChannelLength1 = 32;  % Length of Channel #1
ChannelLength2 = 32;  % Length of Channel #2

Fade1 = .1;           % Rate at which Channel #1 fades
Fade2 = .1;           % Rate at which Channel #1 fades

SNRDB = 15;      % Default SNR dB value for transmitter
%//////// END Parameters for Operating the Transmitter///////////


%//////// Default Parameters for Operating the Reciever       ///////////
% Lets make a good guess as to what the filter should be
GoodGuess = zeros(1,48);
GoodGuess(24) = 1;

% DELTA is synchronization delay
DELTA = 23;

%//////// END Parameters for Operating the Reciever    ///////////

for DD = 1:5

   % Stepsize for DD adaption algorithm
   if DD == 1
      W1delta = .00003;
      W2delta = .00003;
      FEQdelta = .005;
   elseif DD == 2
   % Stepsize for CM adaption algorithm
      W1delta = .00000005;
      W2delta = .00000005;
      FEQdelta = .005;
   elseif DD == 3
   %Stepsize for the MERRY algorithm
      W1delta = .0005;
      W2delta = .0005;
      FEQdelta = .005;
   elseif DD == 4
   %Stepsize for the Trained algorithm
      W1delta = .000003;
      W2delta = .000003;
      FEQdelta = .005;
   elseif DD == 5
```

```
  %Stepsize for the CNA algorithm
    W1delta = .000003;
    W2delta = .000003;
    FEQdelta = .005;
  end

  for Run = 1:10
    index = 0;

  for SNRDB = 15:-1:-10
    index = index + 1;

  [S,r1,r2,RayleighChannel1,RayleighChannel2] = ...
    FSE_Multicarrier_Transmitter(NullCarriers,FFTSIZE,...
    NUMFFT,CP,SNRDB,ChannelLength1,ChannelLength2,Fade1,Fade2);

    if DD == 1
      [BitErrors,MSE,w1,w2] = FSE_Decision_Directed_Receiver(...
        DELTA,GoodGuess,W1delta,W2delta,FEQdelta,S,ChannelLength1,...
        ChannelLength2,NUMFFT,FFTSIZE,CP,r1,r2,DiffEncoding,NullCarriers);
      DecisionDirectedBitErrors(index,Run)=BitErrors;
    elseif DD == 2
      [BitErrors,MSE,w1,w2] = FSE_Constant_Modulus_Receiver(...
        DELTA,GoodGuess,W1delta,W2delta,FEQdelta,S,ChannelLength1,...
        ChannelLength2,NUMFFT,FFTSIZE,CP,r1,r2,DiffEncoding,NullCarriers);
      ConstantModulusBitErrors(index,Run)=BitErrors;
    elseif DD == 3
      [BitErrors,MSE,w1,w2] = FSE_Merry_Receiver(...
        DELTA,GoodGuess,W1delta,W2delta,FEQdelta,S,ChannelLength1,...
        ChannelLength2,NUMFFT,FFTSIZE,CP,r1,r2,DiffEncoding,NullCarriers);
      MerryBitErrors(index,Run)=BitErrors;
    elseif DD == 4
      [BitErrors,MSE,w1,w2] = FSE_Trained_Receiver(...
        DELTA,GoodGuess,W1delta,W2delta,FEQdelta,S,ChannelLength1,...
        ChannelLength2,NUMFFT,FFTSIZE,CP,r1,r2,DiffEncoding,NullCarriers);
      TrainedBitErrors(index,Run)=BitErrors;
    elseif DD == 5
      [BitErrors,MSE,w1,w2] = FSE_Carrier_Nulling_Receiver(...
        DELTA,GoodGuess,W1delta,W2delta,FEQdelta,S,ChannelLength1,...
        ChannelLength2,NUMFFT,FFTSIZE,CP,r1,r2,DiffEncoding);
      CarrierNullingBitErrors(index,Run)=BitErrors;
    end

     pack;

    toc
  end
  end

end
```

Transmitter and Channel

The transmitter and channel have been implemented in the same function call and require a large number of input parameter to effectively control. Most of the input arguments are self-explanatory since they are usually named clearly after what they are named for or are defined in the script file. The output arguments are defined as follows, S is the sequence generated and is necessary for the trained equalizers, r1 and r2 are the sequences that are fed into the receivers, and Rayleigh1 and Rayleigh2 are the Rayleigh channels that were generated when the transmitter was called,

```
%  2LT Nicholas Linnenkamp
%  Blind Adaptive Deconvolution for Multicarrier
%
%  A sequence of 4-QAM data will be created, sequenced and passed through
%  an IFFT, then put into a parallel to serial converter.
%  The resulting signal is transmitted over a channel and noise is added.

function [S,r1,r2,RayleighChannel1,RayleighChannel2] = ...
   FSE_Multicarrier_Transmitter(NullCarriers,FFTSIZE,...
   NUMFFT,CP,SNRDB,ChannelLength1,ChannelLength2,Fade1,Fade2)

%Calculate the Noise Power
SNR = 10^(SNRDB/10);
NoisePower = sqrt(1/SNR);

% Factor to scale the channel power
PowerScalingFactor1 = exp(-Fade1*(0:ChannelLength1-1));
PowerScalingFactor2 = exp(-Fade2*(0:ChannelLength2-1));

% Binary_Sequence is the bits being transmitted over channel grouped in
% an amount appropriate for transmission according to FFTSIZE
S = sign(2*rand(FFTSIZE,NUMFFT)-1)...
   + i*sign(2*rand(FFTSIZE,NUMFFT)-1);

if NullCarriers == 1
   S(1:6,1:NUMFFT) = 0;
   S(FFTSIZE-5:FFTSIZE,1:NUMFFT) = 0;
end

% Now we perform the IFFT to generate output signal
G = sqrt(FFTSIZE)*ifft(S,FFTSIZE);
```

```
% Now we add a cyclic prefix to the data
x(1:CP,1:NUMFFT) = G(FFTSIZE-CP+1:FFTSIZE,1:NUMFFT);
x(1+CP:FFTSIZE+CP,1:NUMFFT)= G(1:FFTSIZE,1:NUMFFT);

% Now we perform parallel to serial positioning of Output_Sequence
x = reshape(x,1,(FFTSIZE+CP)*NUMFFT);

%######### Fading Channel #1 ##########
% Once in serial we convolve it with a random Rayleigh fading channel
RayleighChannel1 = PowerScalingFactor1.*(randn(1,ChannelLength1)...
   +j*randn(1,ChannelLength1));
%RayleighChannel = [RayleighChannel,RayleighChannel];
ChannelOut1 = conv(x, RayleighChannel1);

% Now we add some noise to the channel
r1 = ChannelOut1 + NoisePower*(randn(1,length(...
   ChannelOut1)) + j*randn(1,length(ChannelOut1)));
%######### End Fading Channel #1 ########


%######### Fading Channel #2 ##########
% Once in serial we convolve it with a random Rayleigh fading channel
RayleighChannel2 = PowerScalingFactor2.*(randn(1,ChannelLength2)...
   +j*randn(1,ChannelLength2));
%RayleighChannel = [RayleighChannel,RayleighChannel];
ChannelOut2 = conv(x, RayleighChannel2);

% Now we add some noise to the channel
r2 = ChannelOut2 + NoisePower*(randn(1,length(...
   ChannelOut2)) + j*randn(1,length(ChannelOut2)));
%######### End Fading Channel #2 ########
```

Receiver Code

The following MATLAB code functions are the receiver and decoding portion of the system under test. The receiver portion implements the TEQs and FEQs and outputs the equalized symbols. All receivers assume that the input sequence is differentially encoded and perform differential decoding on the last 500 blocks of symbols (around 20,000 symbols). Since the receiver structure doesn't change it is only appropriate to show how each additional receiver differs from Multicarrier Trained. In order to achieve operation

of the receivers other than Multicarrier Trained replace the respective portion of code in the Multicarrier Trained with the code shown in each additional section.

FSE_Trained_Receiver

```
function [BitErrors, MSE, w1, w2] = FSE_Trained_Receiver(DELTA,...
   GoodGuess,W1delta,W2delta,FEQdelta,S,ChannelLength1,...
   ChannelLength2,NUMFFT,FFTSIZE,CP,r1,r2,DiffEncoding,NullCarriers)

%%%%  -------- start deconvolution process --------  %%%%

w1 = GoodGuess;  %Make a good guess at the inverse filter
w2 = GoodGuess;  %Make a good guess at the inverse filter

% Initialize FEQ here
FEQ = ones(1,FFTSIZE);

% Lets truncate the recieved signals to the shortest of the two
if length(r1) > length(r2)
   r1 = r1(1:length(r2));
else
   r2 = r2(1:length(r1));
end

% Preallocation to speed up computation
Lw1 = length(w1);
Lw2 = length(w2);
h = waitbar(0,'Please wait...Completing Trained Reciever');

y = zeros(1, length(r1)+Lw1);
y1 = zeros(1, length(r1)+Lw1);
y2 = zeros(1, length(r1)+Lw1);

Y = zeros(1, NUMFFT*FFTSIZE);
Err = zeros(FFTSIZE,NUMFFT);
Sout = zeros(1,NUMFFT*FFTSIZE);

FFTBLOCK = 0;

% We run through the incomming sequence sample by sample
for k = 1:length(r1)+Lw1

   % Determine the output y1
   if k <= Lw1
      y1(k) = sum( w1(1,1:k).*r1(k:-1:1));
   elseif k > length(r1)
```

```
   y1(k) = sum (w1(1,k-length(r1)+1:end).*r1(end:-1:...
      end-Lw1+(k-length(r1)+1)));
else
   y1(k) = sum( w1(1,:).*r1(k:-1:k-Lw1+1));
end

% Determine the output y2
if k <= Lw2
   y2(k) = sum( w2(1,1:k).*r2(k:-1:1));
elseif k > length(r2)
   y2(k) = sum (w2(1,k-length(r2)+1:end).*r2(end:-1:...
      end-Lw2+(k-length(r2)+1)));
else
   y2(k) = sum( w2(1,:).*r2(k:-1:k-Lw2+1));
end

% Lets sum the results together
y(k) = y1(k)+y2(k);

% If you have generated enough samples to strip CP off and
% do the FFT then at this point we can update the filter.
if k == (FFTBLOCK+1)*(FFTSIZE+CP)+DELTA

  if FFTBLOCK > 0

     % In order to not run into bounds error stop just before the end
     if FFTBLOCK == NUMFFT - 1
        break
     end

   % We remove the CP by just ignoring it when taking FFT
   Y(FFTBLOCK*FFTSIZE+1:(FFTBLOCK+1)*FFTSIZE) = ...
      sqrt(1/FFTSIZE)*fft(y(FFTBLOCK*(FFTSIZE+CP)+1+DELTA+CP: ...
                            FFTBLOCK*(FFTSIZE+CP)+DELTA+CP+FFTSIZE),FFTSIZE);

   % At this point we continue by processing through the FEQ
   % to determine output sequence Sout.
   Sout(FFTBLOCK*FFTSIZE+1:(FFTBLOCK+1)*FFTSIZE)...
      = Y(FFTBLOCK*FFTSIZE+1:(FFTBLOCK+1)*FFTSIZE) ...
      .*FEQ;

   % At this point we update the adaptive filters based upon Sout.

      %% Trained %%
   % Create error vector

   Err(1:FFTSIZE,FFTBLOCK+1) =
        S(FFTBLOCK*FFTSIZE+1:(FFTBLOCK+1)*FFTSIZE).'...
        -Sout(FFTBLOCK*FFTSIZE+1:(FFTBLOCK+1)*FFTSIZE).';

   FFT = fft(eye(FFTSIZE));

   % Precomputation to reduce size of next block of code
```

```
start = FFTBLOCK*(FFTSIZE+CP)+DELTA+CP;

%#### Updating w1 ########
column = conj(r1(start+1:start+FFTSIZE));
row = conj(r1(start+1:-1:start+1-(Lw1-1)));
Rx = toeplitz(column,row);

for m = 1:FFTSIZE
   %Calculate inner sum
   FFTx = FFT(m,1:FFTSIZE)'*ones(1,Lw1);
   temp = sum(FFTx.*Rx,1);

   deltaW1J(m,1:Lw1) = -2*Err(m,FFTBLOCK+1)*conj(FEQ(m))*temp;
end

% Calculating outer sum
deltaW1Jsum = sum(deltaW1J(1:FFTSIZE,1:Lw1),1);

w1(1,:) = w1(1,:)- W1delta*deltaW1Jsum;
%constrain w1 to unit energy
w1(1,:)= w1(1,:)/sqrt(w1(1,:)...
   *w1(1,:)');
%#### End Updating w1 ####


%###### Updating w2 ######
column = conj(r2(start+1:start+FFTSIZE));
row = conj(r2(start+1:-1:start+1-(Lw2-1)));
Rx = toeplitz(column,row);

for m = 1:FFTSIZE
   %Calculate inner sum
   FFTx = FFT(m,1:FFTSIZE)'*ones(1,Lw2);
   temp = sum(FFTx.*Rx,1);

   deltaW2J(m,1:Lw2) = -2*Err(m,FFTBLOCK+1)*conj(FEQ(m))*temp;
end

% Calculating outer sum
deltaW2Jsum = sum(deltaW2J(1:FFTSIZE,1:Lw2),1);

w2(1,:) = w2(1,:)- W2delta*deltaW2Jsum;
%constrain w2 to unit energy
w2(1,:)= w2(1,:)/sqrt(w2(1,:)...
   *w2(1,:)');
%#### End Updating w2 #####


%###### Updating FEQ ######
deltaFEQJ = -2*Err(1:FFTSIZE,FFTBLOCK+1)...
   .*Y(FFTBLOCK*FFTSIZE+1:(FFTBLOCK+1)*FFTSIZE)';
FEQ(1,:) = FEQ(1,:)-(FEQdelta.*deltaFEQJ).';
%#### End Updating FEQ ####
```

```
      end
      % We update the block so we work on the next one
      FFTBLOCK = FFTBLOCK + 1;
      waitbar(FFTBLOCK/NUMFFT,h)

   end

   % continue determining y(k)
end
% If Differential Encoding has been used then we undo for xmitted sequence
% and for the recieved sequence before determining the number of bit errors

SoutRemoveDiffEncode = zeros(1, NUMFFT*FFTSIZE);
SRemoveDiffEncode = zeros(1, NUMFFT*FFTSIZE);

for k = NUMFFT-500:NUMFFT-3
   for l = 1:FFTSIZE

      % This handles computation of output symbol by removing diff encode
      PhaseOutSymbol = phase(sign(real(Sout(l+k*FFTSIZE)))+ ...
         j*sign(imag(Sout(l+k*FFTSIZE))));
      PhaseNextOutSymbol = phase(sign(real(Sout(l+(k+1)*FFTSIZE)))+ ...
         j*sign(imag(Sout(l+(k+1)*FFTSIZE))));

      if PhaseOutSymbol < 0
         PhaseOutSymbol = PhaseOutSymbol +2*pi;
      end
      if PhaseNextOutSymbol < 0
         PhaseNextOutSymbol = PhaseNextOutSymbol +2*pi;
      end

      % Calculate the Output Symbol
      if PhaseOutSymbol == PhaseNextOutSymbol
         %No Change
         SoutRemoveDiffEncode(l+k*FFTSIZE) = 1 + i;
      elseif PhaseOutSymbol < PhaseNextOutSymbol
         PhaseDiff = PhaseNextOutSymbol-PhaseOutSymbol;
         if PhaseDiff == 0.5*pi
            % 90 Degree Change
            SoutRemoveDiffEncode(l+k*FFTSIZE) = -1 + i;
         elseif PhaseDiff == pi
            % 180 Degree Change
            SoutRemoveDiffEncode(l+k*FFTSIZE) = -1 - i;
         elseif PhaseDiff == 1.5*pi
            % 270 Degree Change
            SoutRemoveDiffEncode(l+k*FFTSIZE) = 1 - i;
         end
      else
         PhaseDiff = 2*pi-(PhaseOutSymbol-PhaseNextOutSymbol);
         if PhaseDiff == 0.5*pi
            % 90 Degree Change
            SoutRemoveDiffEncode(l+k*FFTSIZE) = -1 + i;
```

```
        elseif PhaseDiff == pi
            % 180 Degree Change
            SoutRemoveDiffEncode(l+k*FFTSIZE) = -1 - i;
        elseif PhaseDiff == 1.5*pi
            % 270 Degree Change
            SoutRemoveDiffEncode(l+k*FFTSIZE) = 1 - i;
        end
    end  % End Calculation of Output Symbol

    % This handles computation of input symbol by removing diff encode
    PhaseInSymbol = phase(S(l+k*FFTSIZE));
    PhaseNextInSymbol = phase(S(l+(k+1)*FFTSIZE));

    if PhaseInSymbol < 0
        PhaseInSymbol = PhaseInSymbol +2*pi;
    end
    if PhaseNextInSymbol < 0
        PhaseNextInSymbol = PhaseNextInSymbol +2*pi;
    end

    % Calculate the Input Symbol
    if PhaseInSymbol == PhaseNextInSymbol
        %No Change
        SRemoveDiffEncode(l+k*FFTSIZE) = 1 + i;
    elseif PhaseInSymbol < PhaseNextInSymbol
        PhaseDiff = PhaseNextInSymbol-PhaseInSymbol;
        if PhaseDiff == 0.5*pi
            % 90 Degree Change
            SRemoveDiffEncode(l+k*FFTSIZE) = -1 + i;
        elseif PhaseDiff == pi
            % 180 Degree Change
            SRemoveDiffEncode(l+k*FFTSIZE) = -1 - i;
        elseif PhaseDiff == 1.5*pi
            % 270 Degree Change
            SRemoveDiffEncode(l+k*FFTSIZE) = 1 - i;
        end
    else
        PhaseDiff = 2*pi-(PhaseInSymbol-PhaseNextInSymbol);
        if PhaseDiff == 0.5*pi
            % 90 Degree Change
            SRemoveDiffEncode(l+k*FFTSIZE) = -1 + i;
        elseif PhaseDiff == pi
            % 180 Degree Change
            SRemoveDiffEncode(l+k*FFTSIZE) = -1 - i;
        elseif PhaseDiff == 1.5*pi
            % 270 Degree Change
            SRemoveDiffEncode(l+k*FFTSIZE) = 1 - i;
        end
    end  % End Calculation of Input Symbol

  end  % End for l=
end % End for k
```

```
if NullCarriers == 1
   % Lets determine number of bit errors
Range = FFTSIZE*(NUMFFT-2)-FFTSIZE*313+1:FFTSIZE*(NUMFFT-2);
realBitErrors = sign(real(SoutRemoveDiffEncode(Range)))...
   -sign(real(SRemoveDiffEncode(Range)));
realBitErrors = reshape(realBitErrors,FFTSIZE,313);
realBitErrors = realBitErrors(7:58,1:313);
realBitErrors = reshape(realBitErrors,1,(FFTSIZE-12)*313);

imagBitErrors = sign(imag(SoutRemoveDiffEncode(Range)))...
   -sign(imag(SRemoveDiffEncode(Range)));
imagBitErrors = reshape(imagBitErrors,FFTSIZE,313);
imagBitErrors = imagBitErrors(7:58,1:313);
imagBitErrors = reshape(imagBitErrors,1,(FFTSIZE-12)*313);

BitErrors = sum(sign(abs(realBitErrors)+abs(imagBitErrors)));


MSE = sum(abs(Err.').^2,2)/FFTSIZE;

% Display Scatter plot
%if Display == 1
   DisplaySout = reshape(Sout,FFTSIZE,NUMFFT);
   DisplaySout = DisplaySout(7:58,1:end);
   DisplaySout = reshape(DisplaySout,1,(FFTSIZE-12)*(NUMFFT));
   scatter(real(DisplaySout((FFTSIZE-12)*(NUMFFT-1)-1000:end)),...
      imag(DisplaySout((FFTSIZE-12)*(NUMFFT-1)-1000:end)));
%end

else
% Lets determine number of bit errors
Range = FFTSIZE*(NUMFFT-2)-20000:FFTSIZE*(NUMFFT-2);
realBitErrors = sign(real(SoutRemoveDiffEncode(Range)))...
   -sign(real(SRemoveDiffEncode(Range)));
imagBitErrors = sign(imag(SoutRemoveDiffEncode(Range)))...
   -sign(imag(SRemoveDiffEncode(Range)));
BitErrors = sum(sign(abs(realBitErrors)+abs(imagBitErrors)));


MSE = sum(abs(Err.').^2,2)/FFTSIZE;

% Display Scatter plot
%if Display == 1
   Sout = reshape(Sout,1,FFTSIZE*(NUMFFT));
   scatter(real(Sout((FFTSIZE)*(NUMFFT-1)-1000:end)),...
      imag(Sout((FFTSIZE)*(NUMFFT-1)-1000:end)));
%end
end
close(h)
```

FSE_Multicarrier_Decision_Directed

```
   %% Decison Directed %%
% Create error vector
range = FFTBLOCK*FFTSIZE+1:(FFTBLOCK+1)*FFTSIZE;
DecisionArray = sign(real(Sout(range)))+j*sign(imag(Sout(range)));
Err(1:FFTSIZE,FFTBLOCK+1) =  DecisionArray.' - Sout(range).';

% Generate FFT Twiddle Matrix
FFT = sqrt(1/FFTSIZE)*fft(eye(FFTSIZE));

% Precomputation to reduce size of next block of code
start = FFTBLOCK*(FFTSIZE+CP)+DELTA+CP;


%#### Updating w1 ########
column = conj(r1(start+1:start+FFTSIZE));
row = conj(r1(start+1:-1:start+1-(Lw1-1)));
Rx = toeplitz(column,row);

for m = 1:FFTSIZE
   %Calculate inner sum
   FFTx = FFT(m,1:FFTSIZE)'*ones(1,Lw1);
   temp = sum(FFTx.*Rx,1);

   deltaW1J(m,1:Lw1) = -2*Err(m,FFTBLOCK+1)*conj(FEQ(m))*temp;
end

% Calculating outer sum
deltaW1Jsum = sum(deltaW1J(1:FFTSIZE,1:Lw1),1);

% Update filter
w1(1,:) = w1(1,:)- W1delta*deltaW1Jsum;
%constrain w1 to unit energy
w1(1,:)= w1(1,:)/sqrt(w1(1,:)...
   *w1(1,:)');
%#### End Updating w1 ####


%###### Updating w2 ######
column = conj(r2(start+1:start+FFTSIZE));
row = conj(r2(start+1:-1:start+1-(Lw2-1)));
Rx = toeplitz(column,row);

for m = 1:FFTSIZE
   %Calculate inner sum
   FFTx = FFT(m,1:FFTSIZE)'*ones(1,Lw2);
   temp = sum(FFTx.*Rx,1);

   deltaW2J(m,1:Lw2) = -2*Err(m,FFTBLOCK+1)*conj(FEQ(m))*temp;
end

% Calculating outer sum
deltaW2Jsum = sum(deltaW2J(1:FFTSIZE,1:Lw2),1);
```

```
% Updating Filter
w2(1,:) = w2(1,:)- W2delta*deltaW2Jsum;
%constrain w2 to unit energy
w2(1,:)= w2(1,:)/sqrt(w2(1,:)...
   *w2(1,:)');
%#### End Updating w2 #####


%###### Updating FEQ ######
deltaFEQJ = -2*Err(1:FFTSIZE,FFTBLOCK+1)...
   .*Y(FFTBLOCK*FFTSIZE+1:(FFTBLOCK+1)*FFTSIZE)';
FEQ(1,:) = FEQ(1,:)-(FEQdelta.*deltaFEQJ).';
%#### End Updating FEQ ####
```

FSE Multicarrier Constant Modulus

```
%% Constant Modulus error for TEQ %%
% Create error vector
range = FFTBLOCK*FFTSIZE+1:(FFTBLOCK+1)*FFTSIZE;
Err(1:FFTSIZE,FFTBLOCK+1) =  2 - Sout(range)'.*Sout(range).';

FFT = sqrt(1/FFTSIZE)*fft(eye(FFTSIZE));

% Precomputation to reduce size of next block of code
start = FFTBLOCK*(FFTSIZE+CP)+DELTA+CP;

%#### Updating w1 ########
column = conj(r1(start+1:start+FFTSIZE));
row = conj(r1(start+1:-1:start+1-(Lw1-1)));
Rx = toeplitz(column,row);

for m = 1:FFTSIZE
   %Calculate inner sum
   FFTx = FFT(m,1:FFTSIZE)'*ones(1,Lw1);
   temp = sum(FFTx.*Rx,1);

   deltaW1J(m,1:Lw1) = -4*Sout(range(m)).*real(Err(m,FFTBLOCK+1))*conj(FEQ(m))*temp;
end

% Calculating outer sum
deltaW1Jsum = sum(deltaW1J(1:FFTSIZE,1:Lw1),1);

w1(1,:) = w1(1,:)- W1delta*deltaW1Jsum;
%constrain w1 to unit energy
w1(1,:)= w1(1,:)/sqrt(w1(1,:)...
   *w1(1,:)');
%#### End Updating w1 ####


%###### Updating w2 ######
column = conj(r2(start+1:start+FFTSIZE));
row = conj(r2(start+1:-1:start+1-(Lw2-1)));
```

```
Rx = toeplitz(column,row);

for m = 1:FFTSIZE
   %Calculate inner sum
   FFTx = FFT(m,1:FFTSIZE)'*ones(1,Lw2);
   temp = sum(FFTx.*Rx,1);

   deltaW2J(m,1:Lw2) = -4*Sout(range(m))*real(Err(m,FFTBLOCK+1))*conj(FEQ(m))*temp;
end

% Calculating outer sum
deltaW2Jsum = sum(deltaW2J(1:FFTSIZE,1:Lw2),1);


w2(1,:) = w2(1,:)- W2delta*deltaW2Jsum;
%constrain w2 to unit energy
w2(1,:)= w2(1,:)/sqrt(w2(1,:)...
   *w2(1,:)');
%#### End Updating w2 #####


%###### Updating FEQ ######

%% Decison Directed error for FEQ %%
% Create error vector
range = FFTBLOCK*FFTSIZE+1:(FFTBLOCK+1)*FFTSIZE;
DecisionArray = sign(real(Sout(range)))+j*sign(imag(Sout(range)));
FEQErr(1:FFTSIZE,FFTBLOCK+1) =  DecisionArray.' - Sout(range).';

deltaFEQJ = -2*FEQErr(1:FFTSIZE,FFTBLOCK+1)...
   .*Y(FFTBLOCK*FFTSIZE+1:(FFTBLOCK+1)*FFTSIZE)';
FEQ(1,:) = FEQ(1,:)-(FEQdelta.*deltaFEQJ).';
%#### End Updating FEQ ####
```

FSE Multicarrier Restoration of Redundancy

```
   %% Merry %%
% Create error vector
start = FFTBLOCK*(FFTSIZE+CP)+DELTA+CP;
Err(FFTBLOCK+1) =  y(start)-y(start+FFTSIZE);

%#### Updating w1 ########
% Update filter
w1(1,:) = w1(1,:)- W1delta*Err(FFTBLOCK+1)*(conj(r1(start:-1:start-(Lw1-1)))...
   -conj(r1(start+FFTSIZE:-1:start+FFTSIZE-(Lw1-1))));
%constrain w1 to unit energy
w1(1,:)= w1(1,:)/sqrt(w1(1,:)...
   *w1(1,:)');
%#### End Updating w1 ####


%###### Updating w2 ######
```

```
% Update filter
w2(1,:) = w2(1,:)- W2delta*Err(FFTBLOCK+1)*(conj(r2(start:-1:start-(Lw2-1)))...
   -conj(r2(start+FFTSIZE:start+FFTSIZE+Lw2-1)));
%constrain w2 to unit energy
w2(1,:)= w2(1,:)/sqrt(w2(1,:)...
   *w2(1,:)');
%#### End Updating w2 #####


%###### Updating FEQ ######
%% Decison Directed error for FEQ %%
% Create error vector
range = FFTBLOCK*FFTSIZE+1:(FFTBLOCK+1)*FFTSIZE;
DecisionArray = sign(real(Sout(range)))+j*sign(imag(Sout(range)));
FEQErr(1:FFTSIZE,FFTBLOCK+1) =  DecisionArray.' - Sout(range).';

deltaFEQJ = -2*FEQErr(1:FFTSIZE,FFTBLOCK+1)...
   .*Y(FFTBLOCK*FFTSIZE+1:(FFTBLOCK+1)*FFTSIZE)';
FEQ(1,:) = FEQ(1,:)-(FEQdelta.*deltaFEQJ).';
%#### End Updating FEQ ####
```

FSE Carrier Nulling Algorithm

```
   %% Carrier Nulling %%
% Create error vector
range = FFTBLOCK*FFTSIZE+1:(FFTBLOCK+1)*FFTSIZE;
DecisionArray = zeros(FFTSIZE,1);
Err(1:FFTSIZE,FFTBLOCK+1) =  DecisionArray - Sout(range).';
Err(7:FFTSIZE-6,FFTBLOCK+1) = zeros(1,FFTSIZE-12);


% Generate FFT Twiddle Matrix
FFT = sqrt(1/FFTSIZE)*fft(eye(FFTSIZE));

% Precomputation to reduce size of next block of code
start = FFTBLOCK*(FFTSIZE+CP)+DELTA+CP;

column = conj(r1(start+1:start+FFTSIZE));
row = conj(r1(start+1:-1:start+1-(Lw1-1)));
Rx = toeplitz(column,row);

%#### Updating w1 ########
for m = 1:FFTSIZE
   %Calculate inner sum
   FFTx = FFT(m,1:FFTSIZE)'*ones(1,Lw1);
   temp = sum(FFTx.*Rx,1);

   deltaW1J(m,1:Lw1) = -2*Err(m,FFTBLOCK+1)*conj(FEQ(m))*temp;
end

% Calculating outer sum
```

```
deltaW1Jsum = sum(deltaW1J(1:6,1:Lw1),1);
deltaW1Jsum = deltaW1Jsum + ...
   sum(deltaW1J(FFTSIZE-5:FFTSIZE,1:Lw1),1);

% Update filter
w1(1,:) = w1(1,:)- W1delta*deltaW1Jsum;
%constrain w1 to unit energy
w1(1,:)= w1(1,:)/sqrt(w1(1,:)...
   *w1(1,:)');
%#### End Updating w1 ####


%###### Updating w2 ######
column = conj(r2(start+1:start+FFTSIZE));
row = conj(r2(start+1:-1:start+1-(Lw2-1)));
Rx = toeplitz(column,row);

for m = 1:FFTSIZE
   %Calculate inner sum
   FFTx = FFT(m,1:FFTSIZE)'*ones(1,Lw2);
   temp = sum(FFTx.*Rx,1);

   deltaW2J(m,1:Lw2) = -2*Err(m,FFTBLOCK+1)*conj(FEQ(m))*temp;
end

% Calculating outer sum
deltaW2Jsum = sum(deltaW2J(1:6,1:Lw2),1);
deltaW2Jsum = deltaW2Jsum + ...
   sum(deltaW2J(FFTSIZE-5:FFTSIZE,1:Lw2),1);

% Updating Filter
w2(1,:) = w2(1,:)- W2delta*deltaW2Jsum;
%constrain w2 to unit energy
w2(1,:)= w2(1,:)/sqrt(w2(1,:)...
   *w2(1,:)');
%#### End Updating w2 #####


%###### Updating FEQ ######

%% Decison Directed error for FEQ %%
% Create error vector
DecisionArray = sign(real(Sout(range)))+j*sign(imag(Sout(range)));
FEQErr(1:FFTSIZE,FFTBLOCK+1) =  DecisionArray.' - Sout(range).';

deltaFEQJ = -2*FEQErr(1:FFTSIZE,FFTBLOCK+1)...
   .*Y(FFTBLOCK*FFTSIZE+1:(FFTBLOCK+1)*FFTSIZE)';
FEQ(1,:) = FEQ(1,:)-(FEQdelta.*deltaFEQJ).';
%#### End Updating FEQ ####
```

Bibliography

1. C.R. Johnson, Jr., P. Schniter, I. Fijalkow, L. Tong, J.D. Behm, M.G. Larimore, D.R. Brown, R.A. Casas, T.J. Endres, S. Lambotharan, A. Touzni, H.H. Zeng, M. Green, and J.R. Treichler, The core of FSE-CMA behavior theory, in: S. Haykin, (Ed.), Blind Deconvolution II, Wiley, New York, May, 2000.

2. R.K. Martin, K. Vanbleu, M. Ding, G. Ysebaert, M. Milosevic, B.L. Evans, M. Moonen, and C.R. Johnson, Jr., "Unification and Evaluation of Equalization Structures and Design Algorithms for Discrete Multitone Modulation Systems," *IEEE Trans. on Signal Processing*, vol. 53, pp. 3880-3894, October 2005

3. The European Telecomm. Standards Inst., "Radio Broadcasting System, Digital Audio Broadcasting (DAB) to Mobile, Portible, and Fixed Receivers," ETS 300 401, 1995–1997.

4. The European Telecomm. Standards Inst., "Digital Video Broadcasting (DVB); Framing Structure, Channel Coding and Modulation for Digital Terrestrial Television," ETSI EN 300 744 V1.4.1, 2001 Edition.

5. The Inst. of Electrical and Electronics Engineers, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE Std. 802.11a," 1999 Edition.

6. T. Starr, J.M. Cioffi, and P.J. Silvermann, Understanding Digital Subscriber Line Technology, Prentice Hall PTR, Upper Saddle River, NJ, 1999.

7. K. van Acker, G. Leus, M. Moonen, O. van de Wiel, and T. Pollet, "Per-tone equalization for DMT-based systems," *IEEE Trans. Commun.*, vol. 49, no. 1, pp. 109--119, Jan. 2001.

8. R. K. Martin, J. Balakrishnan, W. A. Sethares, and C. R. Johnson, Jr., "A Blind, Adaptive TEQ for Multicarrier Systems." *IEEE Signal Processing Letters*, vol. 9, no. 11, Nov. 2002, pp. 341-343.

9. D. D. Falconer and F.R. Magee, "Adaptive Channel Memory Truncation for Maximum Likelihood Sequence Estimation," *Bell Sys. Tech. Journal*, pp. 1541-1562, Nov. 1973.

10. J. S. Chow and J. M. Cioffi, "A Cost-Effective Maximum Likelihood Receiver for Multicarrier Systems," in *Proc. IEEE Int. Conf. on Comm.*, June 1992, vol. 2, pp. 948-952.

11. P. J. W. Melsa, R.C. Younce, and C. E. Rohrs, “Impulse Response Shortening for Discrete Multitone Transceivers,” *IEEE Trans. On Comm.*, vol. 44, pp. 1662-1672, Dec. 1996.

12. N. Al-Dhahir and J. Cioffi, “The Combination of Finite-Length Geometric Equalization and Bandwidth Optimization for Multicarrier Transceivers,” in *International conf. on Acoustics, Speech, and Signal Processing*, May 1995, pp. 1201-1204.

13. N. Al-Dhahir and J. Cioffi, “Optimum Finite-Length Equalization for Multicarrier Transceivers,” *IEEE Trans. On Comm.*, vol. 44, no. 1, pp. 56-64, Jan. 1996.

14. S. Haykin, Adaptive Filter Theory, 3rd ed., Englewood Cliffs, NJ: Prentice-Hall, 1996.

15. J. Balakrishnan, R. K. Martin, and C.R. Johnson, Jr., “Blind, Adaptive Channel Shortening by Sum-squared Auto-correlation Minimization” *IEEE Transactions on Signal Processing*, vol. 51, no. 12, pp. 3086–3093. December 2003.

16. Nawaz, R., and J.A. Chambers, “Blind adaptive channel shortening by single lag autocorrelation minimization”, *IEE Electronics Letters*, Vol. 40(25), pp. 1609-1610, 2004.

17. M. de Courville, P. Duhamel, P. Madec, and J. Palicot, “Blind equalization of OFDM systems based on the minimization of a quadratic criterion,” in *Proc. IEEE Int. Conf. on Comm.*, Dallas, TX, June 1996, pp. 1318-1321.

18. F. Romano and S. Barbarossa, “Non-Data Aided Adaptive Channel Shortening for Efficient Multi-Carrier Systems,” in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Proc.*, Hong Kong SAR, China, Apr. 2003, vol. 4, pp. 233--236.

19. T. Miyajima and Z. Ding, “Multicarrier channel shortening based on second-order output statistics,” in *IEEE Trans. Signal Processing Advances in Wireless Communications*, 2003, pp. 145- 149.

20. J.R. Treichler, M.G. Larimore, and J.C. Harp, “Practical blind demodulators for high order QAM signals,” *Proceedings of the IEEE special issue on Blind System Identification and Estimation*, vol. 86, no. 10, pp. 1907-26, Oct. 1998.

21. R. K. Martin, C. R. Johnson Jr., M. Ding, and B. L. Evans, “Exploiting symmetry in channel shortening equalizers,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Apr. 2003.

22. Güner Arslan, Brian L. Evans, and Sayfe Kiaei, “Equalization for Discrete Multitone Transceivers to Maximize Bit Rate”, *IEEE Transactions on Signal Processing*, vol. 49, no. 12, pp. 3123 Dec. 2001

23. Jeremiah F. Hayes, “The Viterbi algorithm applied to digital data transmission,” *IEEE Commun. Mag.*, pp. 26-32, May 2002.

24. B. Sklar, Digital Communications, 2nd ed., Upper Saddle River, NJ: Prentice-Hall, 2000.

Vita

Lieutenant Nicholas Linnenkamp graduated from Antioch Christian School in Arnold, Maryland. He entered undergraduate studies at the University of Maryland in College Park, Maryland where he graduated with a Bachelor of Science degree in Electrical Engineering in May 2003. After working for a while at the United States Patent and Trademark Office, he joined the USAF and was commissioned through Officer Training School at Maxwell AFB in the Fall of 2004.

His first assignment was to National Air and Space Intelligence Center (NASIC) at Wright Patterson AFB, Ohio. Shortly after arriving at NASIC he was given the opportunity to attend the Air Force Institute of Technology as a Watson Scholar and in September 2004, he entered the Graduate School of Engineering. Upon graduation, he will be returning to NASIC.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From – To)*
03-23-2006	Master's Thesis	Jun 2005 – Mar 2006

4. TITLE AND SUBTITLE

Decision Directed and Constant Modulus Algorithms Derived and Evaluated for Multicarrier Systems

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Linnenkamp, Nicholas, L., Second Lieutenant, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GE/ENG/06-36

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory (AFMC)
AFRL/SNRW, Bldg 620
Attn: Mr. James P. Stephens, Sr. DR-III
2241 Avionics Circle Rm N3-F10
Wright-Patterson AFB, OH 45433-7333 TEL: (937) 255-4933 x4239

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The purpose of this research is to derive and examine blind adaptive algorithms for equalizing multicarrier (MC) communication systems and analyze how they perform under varying environmental conditions and parametric variations, focusing on equalizers set in cascade with the channel. Two well-accepted and widely-known cost functions, Decision Directed (DD) and Constant Modulus (CM), are applied to the MC signal structure, and gradient descent algorithms based on both DD and CM functions are derived, analyzed and compared. Comparison of the new algorithms, Multi Carrier Decision Directed (MCDD) and Multi Carrier Constant Modulus (MCCM), focuses on detailing how each algorithm performs when the factors of noise power and symbol synchronization are varied. Additionally, a Frequency Domain Equalizer (FEQ) is developed and employed to de-rotate and resize the output symbol constellation. Both MCDD and MCCM are compared against other blind and trained adaptive MC equalization algorithms in the areas of bit error rate (BER) vs. signal to noise ratio (SNR) and BER vs. synchronization delay. Results are presented showing that MCDD and MCCM perform worse than a MC Trained (MCT) approach but better than both Multicarrier Equalization by Restoration of Redundancy (MERRY) and Carrier Nulling Algorithm (CNA).

15. SUBJECT TERMS
Adaptive Filters, Equalization, Adaptive Communication, Communication, Time Domain Equalization

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
REPORT	**ABSTRACT**	**THIS PAGE**			Richard K. Martin (ENG)
U	**U**	**U**	**UU**	110	**19b. TELEPHONE NUMBER** *(Include area code)* 937-255-3636 ext 4625; e-mail: Richard.Martin@afit.edu